# Bankrupting the Jammer in WSN

Farhana Ashraf, Yih-Chun Hu and Robin H. Kravets
University of Illinois at Urbana-Champaign
Email: {fashraf2, yihchun, rhk}@illinois.edu

***Abstract*—The high vulnerability of the wireless sensor nodes to jamming arises from the low resilience and easy differentiability of protocol control messages, and the high predictability of node wakeup schedules. In this paper, we propose `Jam-Buster` – a jam-resistant solution for WSN, orthogonal to the existing anti-jamming solutions, that increases resilience by using multi-block payloads, eliminates differentiation by using equal size packets and reduces predictability by randomizing the wakeup times of the sensors. While each of these individual components is quite simple, the combination of the three components results in a jam-resilient system that forces the jammer to transmit more enabling faster detection of the jammer by the sensors, and to spend more energy to be effective and so reduce its own lifetime. By modeling our system using game theory and then evaluating the system in a TmoteSky testbed, we show that `Jam-Buster` reduces the overall efficiency of an intelligent jammer.**

## I. Introduction

The combined effect of the broadcast nature of the wireless medium and the limited energy resources available to the sensor nodes makes wireless sensor networks extremely vulnerable to jamming attacks. By interfering with the ongoing transmissions, the jamming signals not only reduce the system throughput by causing packet drops, but also attack the network lifetime by wasting both the senders' and the receivers' energy. Since most existing protocols are designed only for energy-efficiency without considering the presence of a jammer in the system, the jammer can waste a significant amount of the sensors' energy using very few short transmissions, making detection difficult and spending only a little of the jammer's energy [1]. Such an asymmetry of cost enables almost undetectable jammers with limited energy to launch successful attacks against a wireless sensor network.

Several factors contribute to the effectiveness of such cheap yet hard to detect jamming in sensor networks. First, due to the lack of effective error correcting codes, packets have very low *resilience* to interference. Essentially, a jammer only needs to disrupt a few bytes inside a packet to effectively jam the entire packet. The cost for transmitting these few bytes by the jammer are several orders of magnitude lower than the cost of the sender transmitting and the receiver receiving the entire packet. Second, the easy *differentiability* of data and control packets facilitates selective jamming attacks causing the jammer to incur low transmission cost, and yet achieve the same effectiveness as jamming almost all packets. Third, with current periodic MAC protocols, a statistical jammer can easily *predict* future data transmission times. This allows the jammer to duty cycle to avoid unnecessary idle listening costs while still detecting and jamming all packets.

Current solutions to defend WSNs against such cheap jammers are all based on hiding ongoing packet transmissions from reactive jammers. Such jammers must first detect a packet before they can transmit a signal that jams the packet. If a reactive jammer cannot detect a packet, the jammer cannot jam that packet. To achieve this, the sender can try to hide the packets using either a randomized Start-of-Frame Delimiter (SFD) [2] for each packet, channel surfing [3] or a combination of both [2]. However, in both of these techniques, senders and receivers must be tightly synchronized to successfully transmit packets. This results in high overhead even when there is no jammer or the jammer is jamming with a low probability. Moreover, if the jammer has multiple radios and can listen to multiple channels at the same time, or if there are multiple jammers in the neighborhood each listening to a different channel, channel surfing can no longer hide packets from the jammer. Overall, while these techniques may be able to hide some packets from the jammer with similar hardware capabilities, these solutions are quite expensive and have no defense against cheap jamming when the packets are detected.

Instead of trying to outsmart the jammer, we take a completely different approach that attacks the jammer's detectability and energy, and is also orthogonal to the existing solutions. Essentially, our goal is to make the jammer spend more time awake and more time transmitting to successfully jam its target packets, increasing the jammer's detection vulnerability and its energy consumption. The contribution of our research is `Jam-Buster`, a low overhead jam-resistant framework that attacks the three factors that contribute to cheap jamming - resilience, differentiability and predictability. First, `Jam-Buster` uses a *multi-block payload* that improves resilience by enabling the receiver to independently check different blocks inside a packet. As a result, a jammer's disruption is limited to those blocks with which the jamming signal overlaps. Hence, `Jam-Buster` forces a jammer to jam more bytes to successfully jam a whole packet. Second, `Jam-Buster` eliminates differentiability by using the same size packets for both data and control packets, and by randomizing the inter-spacing between the packets. Without differentiability, a jammer does not know the type of a given packet and so is forced to jam all packets that it detects, as opposed to cheaper approaches, such as jamming only control messages. Third, `Jam-Buster` reduces predictability by getting rid of the periodic pattern of wakeup times. Nodes wakeup randomly following a pseudo-random number generator, resulting in random data transmission times and forcing a jammer to remain awake all of the time to detect and jam all packets.

While each component of `Jam-Buster` is quite simple and contributes to increasing the cost of jamming, any component alone only has limited impact on the jammer. For example, resilience only increases the length of the actual jamming

978-1-4673-2433-5/12/$31.00 ©2012 IEEE

signal for packets that the jammer decides to jam. This alone is not effective for detection, since a jammer may only need to jam a few well-chosen packets, or for energy, since the jammer may have good predictability and be able to sleep between transmissions. Similarly, good solutions for both resilience and predictability fail if the jammer has good differentiation. The real benefits come from the combination of the three components, which results in a jammer that must jam all packets using longer jamming signals, incurring high transmission and listening costs as well as forcing the jammer to transmit more often, exposing itself to detection. Since the transmitting sensor nodes and the jammer have contradictory goals, we model their interactions using a game theoretic approach. Our model shows that at the Mixed Strategy Nash Equilibrium (MSNE) of the system, the jammer either transmits an excessive amount of jamming signals, resulting in higher detectability and a very short lifetime, or the jammer duty cycles resulting in longer lifetime but lower effectiveness. Our model shows that `Jam-Buster` reduces the overall efficiency of an intelligent jammer to an inefficient fixed-strategy always-on jammer that jams all packets. We validate our claim by implementing and evaluating `Jam-Buster` in a TmoteSky mote testbed.

One application area of our work is *detection networks in remote areas*, such as at border crossing [4], [5] or in poaching detection [6], [7] or in illegal logging detection. In these applications, sensor nodes are deployed to monitor a large area for intruders and to alert the system if they detect any. To prevent detection, it is important for the intruders to carry jammers to jam these alert messages. However, the jamming signals themselves expose the jammer's location by causing increased packet loss. Hence, in this situation, it is only logical for the intruders to deploy a number of *jamming decoys*, for example mounted on wild animals, to confuse the monitor system of the actual intruder's location. When these decoys jam using sophisticated energy-saving techniques, they can be long-lived and substantially compromise the effectiveness of a remote monitoring system. Because both networks operate in remote areas, both normal users and jammers alike must carry their energy sources into the operating environment, and hence both face limited battery resources. Here, note that the jammers could use bigger batteries than the monitor nodes. However, in our approach, even with larger batteries, long-lived jamming cannot exist, forcing such adversaries to quickly exhaust their energy.

We organize our paper as follows: while section II describes the problems with the current approaches to dealing with the three vulnerabilities, the next three sections discuss the solutions proposed by `Jam-Buster`. In Section VII, we model the interaction between legitimate users and a reactive jammer using game theory. The following two sections describe the implementation details and the results from our implementation. Finally, we conclude the paper in Section X.

## II. Energy-efficient jamming in WSN

Low resilience to jamming signals, easy differentiation of different packet types and high predictability of wakeup schedules — these are the three characteristics that facilitate cheap undetectable yet effective jamming in WSNs. In this section, we discuss the current solutions to jamming attacks in light of these characteristics. Finally, we discuss how we approach these problems as a whole in `Jam-Buster`.

### A. Resilience

Standard communications in WSNs use direct sequence spread spectrum (DSSS) [8] with offset-quadrature phase shift keying (OQPSK) to modulate their signals [9]. Although DSSS protects communication against random noise in the channel, it fails to provide any resilience against intentional jamming since the pseudo-noise (PN) code used by the DSSS in WSN is also known to the malicious nodes. Moreover, due to lack of any packet level protection against errors, the jammer needs to jam only one symbol inside a packet to effectively jam the entire packet. In wireless networks, error correction codes (ECC) [10] provide this packet level protection against errors. However, the low data rate of a WSN channel, the short packet length and the limited resources of sensors make traditional ECC [10], [11], [12], [13] for wireless networks, infeasible for WSN. Hence, a feasible ECC for WSN must be simple yet efficient. Moreover, the level of redundancy provided by the ECC must be adaptive and probabilistic so that a smart jammer can not force the sensors to always use high level of redundancy in their packets and thus exhaust their energy.

Instead of recovering the entire packet, we propose to recover parts of the packet using a low overhead, low complexity mechanism when the packet is partially jammed. We will show that by enabling the amount of that detection to be adapted dynamically, the cost of the jammer can be increased to the point where the jammer's lifetime is significantly reduced while still allowing the sender to recover a significant portion of the transmitted data.

### B. Differentiability

Differentiability facilitates selective jamming [14]. By observing the length and interarrival time between packets, a jammer can easily distinguish the type of a packet, even if the packet content is encrypted. Typically, control packets are shorter than data packets and have smaller inter-packet spacing. In WSN, the reactive jammer knows the packet length even before the radio finishes the packet transmission since the `CC2420` radio transceiver attaches the packet length information at the beginning of every packet. Given knowledge of the message type and the specific protocol being used, a jammer can target to prevent the transmission of data packets in the first place by CTS corruption jamming, or to force the sender to retransmit the packet multiple times until the sender gives up and drops the packet from its queue by applying DATA and ACK corruption jamming.

By removing differentiability, knowledge of the specific MAC protocol no longer benefits the jammer, which now must decide to jam packets without knowing the type of the packet. We will show that by eliminating the correlation between packet type and inter-frame spacing as well as the correlation between packet size and packet type, the jammer's cost is again significant increased.

### C. Predictability

By observing the temporal arrangement of packets induced by the nature of the MAC protocol, a jammer can easily identify a pattern in the data transmission times. Once the pattern is detected, an energy-efficient jammer wakes up only during the predicted data transmission times and thus saves idle listening costs by sleeping during the rest of the time. Obviously, the pattern detection algorithm [15] used by the jammer will be different for different MAC protocols since the temporal organization of medium accesses varies with MAC protocols. However, the patterns are always created either due to the periodic send of the senders or the periodic listen of the receivers, both of which are induced by the periodic duty-cycle of the nodes in the network.

The key to reducing this vulnerability is to break any existing pattern between the successive wakeup times. We will show that an intelligent MAC protocol can use random wakeup times to effectively cause the jammer to either stay awake all of the time or to miss a significant number of packets.

### D. Jam-Buster

While each of the above three problems are quite simple and have simple defenses, defending against any one of them alone is not sufficient to inflate the cost of the jammer. Improving resilience increases the costs of the jammer's transmissions, eliminating differentiation increases the number of packets that a jammer needs to jam and eliminating predictability causes the jammer to spend more time idle listening. To this end, we have designed `Jam-Buster`, a jam resilient framework for MAC protocols that provides improved performance and reduced energy consumption in the face of a reactive jammer, while causing the jammer to significantly increase the number of jamming signal transmissions facilitating jammer detection and so limit its impact, lifetime or both. In the next three sections, we discuss how `Jam-Buster` tackles these problems using a *multi-block payload* to improve resilience, *look-alike packets* to eliminate differentiability and *random wakeup* to eliminate predictability.

## III. Multi-Block Payload: Defense against Low Resilience

`Jam-Buster` provides a level of protection inside each packet that is of low cost to the sender but of high cost to the jammer to jam. Instead of using a single CRC at the end of each packet like traditional 802.15.4 packets, the sender includes additional checks throughout the packet. The benefits of the multiple checks is twofold. First, these checks help the receiver to determine if there is any valid data in the packet. Second, these checks force the jammer to jam more bytes within each packet to cause the entire packet to be discarded.

`Jam-Buster` divides the payload inside each packet into multiple blocks where each block has its own CRC. We refer to this packet structure as *Multi-Block Payload Packet*. Figure 1 shows the structure of a 3-block payload packet. For a reactive jammer to jam a packet, the jammer needs to switch its radio to the transmission mode, after detecting the Start-of-Frame-Delimiter (SFD) of the ongoing transmission. In our experiments as described in Section IX, we observed that due to the turn around time to switch to transmission mode, the reactive jammer could never jam the header of the ongoing packet. As a result, we have chosen not to repeat the header for each block. However, if this were not the case, we would need to repeat certain header fields, in each block.

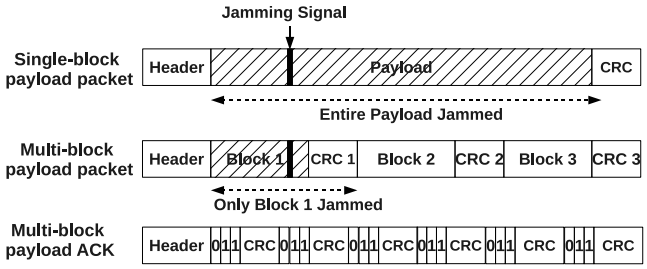


Fig. 1. Packet format of a multi-block payload packet

The per-block CRCs enable the receiver to independently verify each block for corruption. After receiving the packet, the receiver marks a block inside the packet to be *jammed* only if the CRC for that particular block fails, and marks the block as *unjammed* otherwise. In Figure 1, we marked the jammed portion of a packet as hatched areas. We observe from this example that the short jamming signal jams the entire payload of the traditional 802.15.4 packet, whereas it can only jam the first block of the 3-block payload packet. The data bytes in the rest of the blocks remain unjammed.

Since the receiver can successfully receive some of the blocks inside the packet when the jamming signal only partially interferes with the packet, the traditional acknowledgement indicating the receipt of the entire packet is no longer adequate. Rather, the receiver must specifically acknowledge the receipt of each successfully received block inside the packet. Instead of using separate acknowledgement packets for each block, the receiver uses a single acknowledge packet for the entire packet where a bitmap represents the success or failure of the reception of successive blocks inside the packet. Figure 1 shows the structure of an acknowledgement packet to a 3-block payload data packet. The $\{0, 1, 1\}$ bitmap of the ACK response indicates to the sender that the receiver successfully received block 2 and 3, whereas failed to successfully decode block 1. Hence, the selective ACK allows the sender to selectively retransmit the blocks that were jammed.

For multi-block payloads to be successful, it is extremely important to choose the right number of blocks inside the packet. With more blocks, *multi-block payload* provides finer grained protection from jamming. However, as the number of blocks increases, the overhead due to CRC also increases. Since WSN packets are small, with more blocks, the CRC overhead could overwhelm any benefit achieved due to multi-block payload packets. In Section VII, we discuss in detail how `Jam-Buster` chooses the optimal number of blocks in its packets.

## IV. Look-alike Packets: Defense against Differentiability

Packet size and inter-frame spacing are two key indicators of different packet types. Jam-Buster uses two techniques to make all packets *look-alike*. First, all packets are the same size, regardless of the amount of information or the packet type. Second, inter-frame spacing is randomized to prevent the jammers from using communication patterns to determine packet type. To complete the solution, all packets are encrypted to hide the type information inside the packet header.

Generally, control packets are shorter and data packets are longer. Jam-Buster has two options to make the control and the data packet sizes equal: either break each long packet into multiple short packets, or pack each short packet inside a long packet. Since each packet needs its own header, using multiple short packets incurs an increased header overhead. In contrast, the overhead due to longer packets incur because the control packets do not have sufficient information to fill up the inflated payload. However, with *multi-block payload packet*, discussed in Section III, the sender can intelligently use this extra space to improve the jam-resilience of its control packets. In Figure 1, we provide an example of applying a combination of both *multi-block payload* and *look-alike packets* to an ACK packet. The same block is repeated multiple times to fill up the entire payload to provide high jam-resilience as well as to eliminate differentiability. Due to this increased protection, Jam-Buster chooses to use longer packets for both its control and data packets. The only downfall of this approach is the overhead paid by the long control packets when there is no jammer present in the network. But this overhead is not significant in a WSN environment since packets in a typical sensor networks are really small. Even the maximum payload size in WSN is limited to 80 bytes. However, we understand that the overhead will be significant in a 802.11 network since 802.11 data packets are large compared to the control packets. If Jam-Buster was to be used for 802.11 networks, then such overheads could be reduced by using cumulative ACKs or by eliminating the need for ACKs altogether by using erasure coding. However, all these mechanisms are out of the scope of this paper, and we are only interested in Jam-Buster's applicability in a WSN scenario. When a jammer is present in the network, the overhead due to long packets becomes minimal compared to the benefits achieved from more jam-resilient control packets.

Jam-Buster uses a different random inter-frame spacing, RIFS, between each of its packets belonging to a single transmission. If $\texttt{IFS}_{min}$ is the minimum turn around time required to switch the radio from the reception to the transmission state, and $\texttt{IFS}_{max}$ is the maximum allowable inter-frame spacing (i.e., $\texttt{IFS}_{max} = \texttt{EIFS}$), then the value of RIFS is chosen from the following random number generator:

$$\texttt{RIFS} = \texttt{random}\ (\texttt{IFS}_{min}, \texttt{IFS}_{max}).$$

This implies that when a receiver receives a packet, it waits for a randomly chosen RIFS before sending its ACK. On the other hand, since the sender has no knowledge about the RIFS chosen by the receiver, the sender waits for an $\texttt{IFS}_{max}$ time duration before finally considering the ongoing transmission to be unsuccessful. In addition to that, the sender must also wait for at least an $\texttt{IFS}_{max}$ duration before starting its own transmission in order to make sure the previous transmission has been completed. Hence, $\texttt{DIFS} > \texttt{IFS}_{max}$. The overhead of using RIFS is a slightly increased delay due to the longer waits between packet transmissions. However, our model in Section VII shows that the increase is minimal and Jam-Buster benefits from *look-alike packets* due to more successful transmissions resulting in lower overall delay.

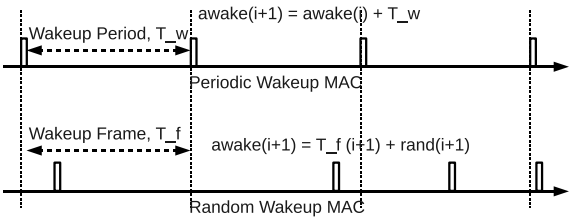


Fig. 2. Random wakeup

## V. Random Wakeup: Defense against High Predictability

Existing MAC protocols have periodic wakeup schedules to save energy. An attacker can listen to determine the period of a node's wakeup schedule, and then can jam that one node and still sleep just as much as that node. To create asymmetry between the jammer and a legitimate network user, Jam-Buster randomizes the wakeup time of each sender, making it unpredictable to the jammer.

Each node divides its time into equal sized slots called *wakeup frames*, $T_f$ (see Figure 2). These wakeup frames are analogous to the wakeup periods in a periodic MAC protocol. However, with random wakeups, a node wakes up at a random time within each wakeup frame. The offsets $r_n$ from the start of the wakeup frames are determined from a pseudo-random number generator PRNG - the seed of which is unique for each node. The combined effect of the unsynchronized start of the wakeup frames for different nodes and the different offsets for the different wakeup frames, results in a unique wakeup pattern for each node in the network.

Using random wakeups, Jam-Buster eliminates predictability in two ways. First, due to the random offset of the wakeup time, $r_n$, in each slot, the jammer can no longer predict future wakeup time of a node by correlating the temporal arrangements of the previous wakeup times of the same node. Second, because each node has a unique seed, the jammer can no longer predict the future behavior of a node by watching the behavior of another node in the same network. In this situation, the only way for the jammer to predict a node's future wakeup times is to have exact knowledge of the wakeup parameters of that node, i.e., the seed of that node's PRNG, length of $T_f$ and the clock offset between the jammer and the sensor node, $\texttt{offset}_{S-J}$. Given this information, the jammer can determine node $A$'s current wakeup frame number, $\hat{n} = \lceil \frac{t+\texttt{offset}_{S-J}}{T_f} \rceil$, at time $t$ and then the exact wakeup time,

$\hat{a_n}$, in that wakeup frame:

$$\hat{a_n} = (\hat{n} - 1).T_f + (r_{\hat{n}} \ \texttt{mod} \ T_f) - \texttt{offset}_{S-J}.$$

However, whenever these wakeup parameters are exchanged between nodes, the packets are encrypted using a key not known to the jammer. Hence, we can safely assume that the jammer cannot obtain the wakeup parameters and hence cannot generate the predicted wakeup times.

Since the jammer can no longer predict the future transmission times with random wakeups, the only way for a reactive jammer to detect and jam all packets is to remain awake all of the time. However, such an *always-on jammer* will have a very short lifetime due to its high idle listening costs, and hence will not be very effective. Our model in Section VII validates this claim. The other option for the jammer is to duty cycle and be effective for a longer time, but fail to detect any transmission that occurs while it is asleep. Thus, the jammer has to make a tradeoff, between a higher duty cycle to increase the number of packet detections and hence successful jamming, and a lower one to save energy and extend the jammer's lifetime. In Section VII, we will show that in a network with random wakeups, the *duty-cycled* approach is dominated by an *always-on* strategy since the *duty-cycled* strategy results in more missed opportunities for the jammer to jam packets.

## VI. Coordination with Random Wakeup

The biggest challenge of using random wakeups is to coordinate the sender-receiver pairs to ensure successful data transmissions. An ideal coordination mechanism must be *jam-resilient* as well as *energy-efficient*. If the coordination itself is not jam-resilient, the jammer can exploit this vulnerability to pose energy-efficient jamming attacks, essentially nullifying the benefits obtained due to random wakeups. In this section, we analyze the existing MAC protocols to identify which protocols are appropriate to coordinate the nodes in a system using `Jam-Buster`.

The simplest way to achieve coordination is to use any existing random wakeup MAC protocol, e.g., RAW [16], PW-MAC [17]. In both protocols, coordination is achieved by transmitting beacons at the beginning of each wakeup. This incurs extremely high overhead at low traffic. As a result, such protocols are not appropriate for the diverse traffic expected in detection networks. However, if the network expects high traffic, `Jam-Buster` can adopt any of these protocols .

For diverse traffic, `Jam-Buster` can choose an appropriate periodic MAC protocol (*base* protocol) and adapt it to support random wakeups. Typically, hybrid protocols, such as NPM [18] and Wise-MAC [19], are the most energy-efficient solutions to coordinate nodes with periodic wakeups and diverse traffic. These protocols use a combination of *signaling* and *synchronization* to balance the signaling cost at high traffic and the synchronization overhead at low traffic. While signaling allows a sender to communicate with its receiver when the receiver's recent wakeup information is not available, synchronization helps to keep the signal lengths shorter by utilizing estimates about the receiver's wakeup times. To coordinate nodes with random wakeups using these periodic hybrid protocols, the *base* protocols require different modifications for their signaling and the synchronization components.

We can derive the required modifications from the two major differences that exist between periodic and random wakeups. First, the duration between two successive wakeups of a node with periodic wakeup is always $T_f$, whereas with random wakeup, this duration can vary between $(0, 2T_f)$. Hence, the sender may need to preamble for a maximum duration of $2T_f$ to make sure it successfully signals the receiver. Moreover, the sender must send short preambles during this duration instead of sending long preambles to prevent the jammers from having a perfect predictability due to the long preambles. Second, only $T_f$ is no longer enough to predict future wakeup times of the receiver. With random wakeups, the sender must have additional information about the seed of the receiver's `PRNG`, and the clock offset between the sender $S$ and the receiver $A$, $\texttt{offset}_{A-S}$. To reduce the effect of clock drift on the timing information, each node sends its next slot number $n$ and the offset to the start of its $n$-th wakeup frame, $\texttt{offset}_n$. If $S$ receives this information at time $t_1$, then $S$ can easily determine node $A$'s slot number, $m = n + \lfloor \frac{t_2 - t_1 - \texttt{offset}_n}{T_f} \rfloor$ and from that the wakeup time, $a_m$ at any given time $t_2$:

$$a_m = (t_1 + \texttt{offset}_n) + (m - n).T_f + (r_m \ \texttt{mod} \ T_f).$$

Hence, to enable random wakeup, the *base* protocol must be modified such that each node includes the additional wakeup information inside its synchronization packets, and the sender sends its data packets at the calculated wakeup time, $a_m$. To provide protection against the jammer, all this information will be encrypted inside the packet. This implies modifying the ACK packets of Wise-MAC [19] and the CTRL packets of NPM [18] to include random wakeup information.

In summary, `Jam-Buster` is not restricted to using any specific MAC protocol for coordinating its sender-receiver pairs. Essentially, `Jam-Buster` can use any existing random wakeup protocol in case of a high traffic scenario. In case of a low traffic or diverse traffic scenario, Jam-Buster can be layered on any appropriate periodic MAC protocol; that is, one that uses short preambles and estimates the receiver's wakeup time to shorten its preamble duration.

## VII. Analytic Model of Jam-Buster

In this section, we model the interaction between a legitimate user that uses `Jam-Buster` and an intelligent jammer to find out an upper bound on the best performance of the jammer. The model essentially validates `Jam-Buster`'s feasibility by showing that the upper bound lowers significantly when the regular nodes use `Jam-Buster` as their defense. Table I lists the notations used to explain the model. Moreover, the analysis also provides guidelines on the optimal number of blocks to be used by the sensors and the optimal strategy to be used by the jammer in terms of choosing its duty cycle and jamming signal length.

### A. Interdependence between Strategies

The regular nodes and the jammer have two completely opposite goals. While `Jam-Buster`'s primary goal is to

| Variable | Meaning | Values |
|---|---|---|
| $k$ | number of blocks | varying |
| $\ell_{jam}$ | jamming signal length | varying |
| $L$ | maximum payload | 80 B |
| $\ell_{crc}$ | CRC length | 6 B |
| $\ell_{data}$ | data bytes per block | |
| $b_{jammed}$ | jammed blocks | |
| $\texttt{DATA}_{\texttt{unjammed}}$ | unjammed bytes | |
| $T_f$ | wakeup frame | 1 msec |
| $T_a$ | awake interval | 100 msec |
| $\texttt{d}_{jammer}$ | jammer's duty cycle | varying |
| $\texttt{d}_{RN}$ | node's duty cycle | 1% |
| $T_d$ | packet inter-arrival time | varying |
| $P_i$ | idle power | 1.278 uJ |
| $P_t$ | transmission power | 52.2 uJ |
| $P_r$ | receive power | 59.1 uJ |
| $x$ | relative battery level | varying |
| $\texttt{rate}_c$ | channel rate | 250 Kbps |

TABLE I
LIST OF NOTATIONS FOR THE MODEL

maximize the total number of bytes successfully received at the receiver, $\texttt{DATA}_{\texttt{unjammed}}$, the jammer's goal is to minimize the same by jamming the packets. $\texttt{DATA}_{\texttt{unjammed}}$ essentially depends on both the number of blocks, $k$, used by the sender in that *multi-block payload* packet and the length of the jamming signal, $\ell_{jam}$. While an increased $k$ benefits the sender by reducing the average number of jammed blocks in a packet, $b_{jammed} = \texttt{min}(\lceil \frac{\ell_{jam}}{\ell_{data}+\ell_{crc}} \rceil + 1, k)$, larger $k$ also has the negative effect of reducing the number of data bytes inside each block, $\ell_{data} = \lfloor \frac{L}{k} \rfloor - \ell_{crc}$. This results in $\texttt{DATA}_{\texttt{unjammed}}$ to be extremely sensitive to the choice of $k$.

$$\texttt{DATA}_{\texttt{unjammed}} = (k - b_{jammed}).\ell_{data}.$$

Choosing the optimal $k$ becomes even more complicated as $\texttt{DATA}_{\texttt{unjammed}}$ depends on $b_{jammed}$, which in turn depends on $\ell_{jam}$. Moreover, there is no way for the regular node to obtain pre-knowledge about the $\ell_{jam}$ signal to be used by the jammer. An intelligent jammer can exploit this fact and change its $\ell_{jam}$ signals such that the sender can no longer maximize its $\texttt{DATA}_{\texttt{unjammed}}$ by using an optimal $k$ all of the time. Since a fixed strategy is no longer adequate, we model the entire interaction using a game theoretic approach and determine a mixed strategy for choosing $k$ and $\ell_{jam}$ such that neither the sender nor the jammer can improve its payoff by changing the strategy. The $\texttt{DATA}_{\texttt{unjammed}}$ at this Mixed Strategy Nash Equilibrium (MSNE) state essentially determines the upper bound of the jammer's performance.

### B. Modeling Jam-Buster using Game Theory

The game has two parties: a regular node, $RN$, that uses Jam-Buster and an intelligent jammer, $J$. Formally, we define the game as $(S, f)$, where $S$ is the set of strategy profiles (i.e., $S = S_{RN} \times S_J$) and $f$ is the set of payoffs (i.e., $f = f_{RN} \times f_J$) of both the parties. Here, the sender's strategy $S_{RN}$ is to choose $k$, whereas the jammer's strategy $S_J$ is to choose $\ell_{jam}$.

*Payoff Functions:* The sensor's payoff $f_{RN}$ is determined by the total unjammed data bytes that the node receives during its lifetime. While $\texttt{DATA}_{\texttt{unjammed}}$ captures the total unjammed data bytes from a particular packet, it does not capture the relative energy-constraint of the jammer and the regular nodes in the network. Since the jammer can only jam until its own lifetime expires, we need to incorporate the energy constraints into $f_{RN}$. Finally, the jammer's goal is to minimize $f_{RN}$. Hence, the interaction between the jammer and the nodes can be represented as a zero-sum game, and the jammer's payoff is defined as: $f_J = -f_{RN}$.

$f_{RN}$ is a function of $\texttt{DATA}_{\texttt{unjammed}}$ and the relative lifetime, life, which in turn depends on the energy spent by the nodes and the jammer, $E_{RN}$ and $E_J$ respectively, and the relative battery level, $x$, of the jammer and the nodes:

$$\texttt{life} = \texttt{max}(\frac{E_j}{x.E_{RN}}, 1).$$

Both $E_{RN}$ and $E_J$ depend on the traffic scenario, the network density and the adopted duty cycles. We assume a single-hop network with $n$ regular nodes and 1 jammer where each sensor generates traffic at $T_d$ inter-arrival times and the jammer's target is to jam these transmissions. We limit our analysis to a single-hop network because the jammer can only jam those transmissions that occur within its one hop distance. All nodes wakeup randomly maintaining a duty cycle of $d_{RN}$. Hence, we determine $E_{RN}$ within one $T_d$ period as follows:

$$E_{RN} = \frac{L}{\texttt{rate}_c} P_t + T_d.\texttt{d}_{RN}.P_i.$$

We assume that jammer joins the network with a battery $x$ times larger than that of a legitimate node. In our model, we keep $x$ as a parameter and vary $x$ from 1 to 100. This allows us to simulate the effect of multiple jammers present in the network with just 1 jammer. Essentially, we can consider the scenario with 2 jammers with each one having $\frac{x}{2}$ times larger battery than a legitimate node using the same single jammer scenario. We can also simulate jammers with unlimited energy resources with a very large $x$ value. We assume that the jammer's duty cycle is $\texttt{d}_{jammer}$. In our model, an *always-on jammer* is represented with $\texttt{d}_{jammer} = 100\%$, while a no jammer situation is represented with $\texttt{d}_{jammer} = 0\%$. A jammer with $\texttt{d}_{jammer}$ duty cycle, can detect only $\texttt{d}_{jammer}$ portion of all packets. Hence, the jammer spends $(n.\texttt{d}_{jammer}.\ell_{jam}/\texttt{rate}_c)$ time transmitting to jam all $n.\texttt{d}jammer$ packets detected from $n$ nodes. We calculate $E_J$ as follows:

$$E_J(\ell_{jam}) = \frac{n.\texttt{d}_{jammer}.\ell_{jam}}{\texttt{rate}_c} P_t + T_d.\texttt{d}_{jammer}.P_i,$$

Finally, we incorporate both life and $\texttt{DATA}_{\texttt{unjammed}}$ to obtain $f_{RN}$. Since, the jammer can detect and jam a packet with $(1 - \texttt{d}_{jammer}/\texttt{life})$ probability, $f_{RN}$ is as follows:

$$f_{RN} = \frac{\texttt{d}_{jammer}}{\texttt{life}} \texttt{DATA}_{\texttt{unjammed}} + (1 - \frac{\texttt{d}_{jammer}}{\texttt{life}})(L - k\ell_{crc}).$$

Here, we can see that when the jammer has unlimited energy resources, i.e., $\texttt{life} = 1$, and the jammer does not duty cycle, i.e., $\texttt{d}_{jammer} = 100\%$, the node's average payoff is simply the unjammed bytes from a packet, $\texttt{DATA}_{\texttt{unjammed}}$.

*Solving the MSNE:* To solve the MSNE, we calculate $f_{RN}$ and $f_J$ for all ($k$, $\ell_{jam}$) values at different $\mathtt{d}_{jammer}$ settings, and then input these values to our MSNE solver written in `matlab`. The solution assigns probabilities to each strategy inside $S_{RN}$ and $S_J$. If MSNE assigns probability $p_i$ to the $i$-th strategy of $S_{RN}$, then `Jam-Buster` follows that strategy with probability $p_i$. The same is the case for the jammer.

The parameter values used in our `matlab` simulations are listed in Table I. We varied the traffic by choosing different $T_d = \{5, 10, 60, 120\}$ `sec` values and the network density by setting $n = \{1, 2, 3, 4\}$. We changed the jammer's setting by setting $x = \{1, 2, 5, 10, 25, 50, 75, 100\}$ and $\mathtt{d}_{jammer} = \{0\%, 25\%, 50\%, 75\%, 100\%\}$. Interestingly, for all the settings, the MSNE solution indicated that the nodes must always use different $k$ values with equal probabilities and in response the jammer must always send the longest $\ell_{jam}$ to maximize their payoff at the equilibrium state. The graphs are ommitted due to space constraints. For very high relative battery levels ($x > 80$), MSNE suggests the nodes to use the higher $k$ values with slightly increasing probability. However, the jammer's strategy remains the same.

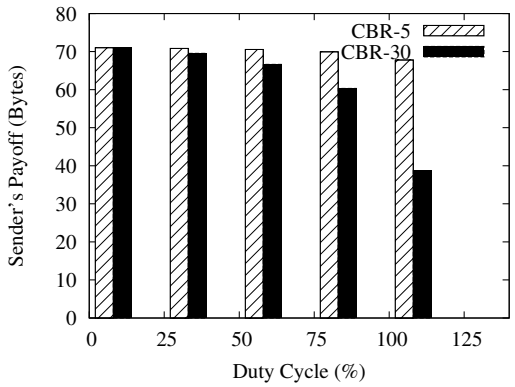


Fig. 3. $f_{RN}$ for different $T_d$

Next, we need to determine the jammer's duty cycle level at the equilibrium. We observe that irrespective of the current $k$ used inside the packet, $f_{RN}$ is the minimum when the jammer is always-on, i.e., $\mathtt{d}_{jammer} = 100\%$ (graph ommitted due to space constraints). This is expected from the $f_{RN}$ equation since the jammer can now detect and jam all packets. For high traffic rates, we observe the same payoffs for both the *always-on jammer* and the *duty-cycled jammer* (see Figure 3). However, the situation is not the same for low traffic. The jammer then spends more time idle listening to detect packets rather than actually jamming packets. Hence, as the traffic inter-arrival time increases, the sender achieves lower payoff with higher $\mathtt{d}_{jammer}$ jammers, and the lowest when the jammer is always awake. At this point, the sender achieves only a $50\%$ payoff which maximizes the jammer's payoff. Thus, the MSNE suggests that regardless of the shorter lifetime, the jammer should always remain awake to achieve maximum payoff.

## VIII. Implementation Details

We implemented `Jam-Buster` in TinyOS, and ran the experiments on `Tmote Sky` motes.

### *Implementing `Jam-Buster`*

For all sensors in the network, we implemented the three components of `Jam-Buster`. While implementing look-alike packets and random wakeups was easy, the multi-block payload required modifications to the radio transceiver and the `CC2420` driver of the `Tmote Sky` motes. The typical operation of a transceiver is to check the CRC of the packet and pass the packet to the application layer only if the CRC passes. To enable the receiver to independently check each block, we disabled the default CRC check enabling the `CC2420` transceiver to pass the packet to the upper layer regardless of whether it passes or fails the CRC check. The application layer then checks individual blocks inside a packet for interference.

### *Implementing the reactive jammer*

A reactive jammer transmits its jamming signal upon detecting an ongoing transmission. In the default setting, the `CC2420` transceiver backs off its transmission when it finds a busy channel using a mechanism called clear channel assessment (CCA). While implementing our reactive jammer, to ensure that the jamming signal interferes with the ongoing packet, the jammer turns off its default CCA before transmitting its jamming signal. Furthermore, the jammer must start transmitting very fast to ensure that the jamming signal coincides with the ongoing data transmission. We observed that transmitting a packet is a two-step process. At first, the `CC2420` transceiver loads the packet inside its `TXFIFO`. Then the transceiver starts transmitting the packet into the radio channel. From our experiments, we observed that loading a packet inside the `TXFIFO` requires around $3-4$ `msec`. Within this time, the ongoing packet transmission is over. To avoid this delay due to loading time, we preload the `TXFIFO` with a jamming packet, and the `CC2420` chip starts transmitting the packet as soon as it detects the packet. To send the packet loaded inside the `TXFIFO` and avoid the CCA, we use the `CC2420Transmit.resend(FALSE)` function.

The reactive jammer listens to the channel for an SFD to detect an ongoing transmission. We implement the detection by attaching an interrupt to the SFD pin of the `CC2420` chip on the jammer. It is important to stop the SFD interrupt on the SFD of the jamming signal. We implemented two jammers: `jam-1` which sends 1 pulse when it detects a packet, and `jam-2` which sends 2 back-to-back jamming pulses to jam a packet. Due to hardware constraints, there is 11 bytes of unjammed data between the 2 pulses of `jam-2`.

## IX. Experimental Evaluation

The goal of our evaluation is to show that `Jam-Buster` can successfully improve the jam resilience of wireless sensor networks: (1) by forcing any jammer to transmit more jamming signals enabling faster jammer detection, and (2) by

forcing an energy-constraint jammer to spend more energy jamming and thus bust the jammer. Our evaluation also proves the feasibility of our proposed solutions by showing that Jam-Buster incurs only minimal overhead when there is no jammer in the system.

*Evaluation Metrics*

We evaluate Jam-Buster's effectiveness against two types of jammers: *always-on jammer* and *duty-cycled jammer*. For both jammers, we use the *average unjammed bytes per packet*, $\texttt{DATA}_{\texttt{unjammed}}$ to evaluate Jam-Buster's effectiveness. A high $\texttt{DATA}_{\texttt{unjammed}}$ represents better jam-resilience of the system, while a low $\texttt{DATA}_{\texttt{unjammed}}$ represents a more effective jammer.

For an energy-constraint jammer, we use *time-average unjammed bytes*, $f_{RN}$ to capture the overall effectiveness of the senders and the jammer given their energy constraints. Similar to $\texttt{DATA}_{\texttt{unjammed}}$, a high $f_{RN}$ indicates Jam-Buster's effectiveness while a low $f_{RN}$ indicates the jammer's effectiveness. To capture the effect of relative battery levels of the jammer and the sender, $x$ using our experiments, we evaluate *ratio of the battery exhaustion rate of the jammer to the sender*, life. When $x = 1$, life represents the relative lifetime of each sender compared to the jammer. A high life indicates the Jam-Buster's effectiveness in exhausting the jammer's energy resources.

*Network Setup and Traffic Scenario*

Our network is a single hop network with $n$ senders sending CBR traffic to a sink, and 1 jammer. We vary $n$ from 1 to 4 to evaluate the effect of multiple flows in the network. To capture the effect of traffic generation rates, we varied the CBR inter-arrival times starting from 5 sec to 30 sec in steps of 5 sec. All senders wake up randomly maintaining a $1\%$ duty cycle. We vary the jammer's duty cycle from $100\%$ to $0\%$ in steps of $25\%$. While $100\%$ duty cycle means that the jammer is always awake and thus can detect and jam all packets, $0\%$ duty cycle means that the jammer is never awake representing a no jammer scenario. To capture the effect of jamming signal length, we evaluate our system with jam-1 and jam-2 jammers. In our evaluations, $k = 1$ represents the scenario when the nodes have no defense against the jammer, and $k_{rand}$ represents Jam-Buster with nodes running *multi-block payload* with equal probabilities for all blocks, 1 to 7. Finally, $k_{opt}$ represents the optimal scenario, where each node has the exact knowledge about the jammer's strategy, and thus can choose the optimal number of blocks for each packet to achieve maximum throughput. We want to emphasize that $k_{opt}$ is not feasible scenario with smart jammers, and is determined in this case using extensive experimentations. The results in this section are obtained from the average of 5 runs, where we ran each experimental setup for 30 minutes.

*Jam-Buster vs. Always-on Jammer*

Our goal is to show that Jam-Buster improves jam-resilience against an always-on jammer by increasing

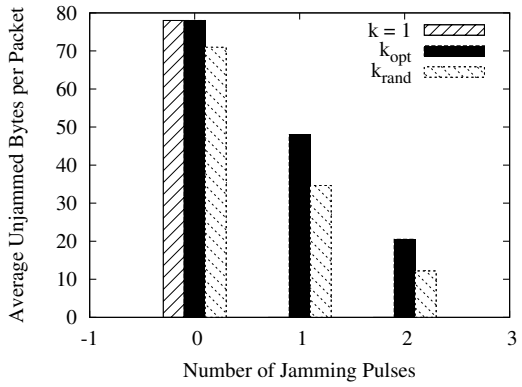


Fig. 4. Unjammed bytes per packet with always-on jammer

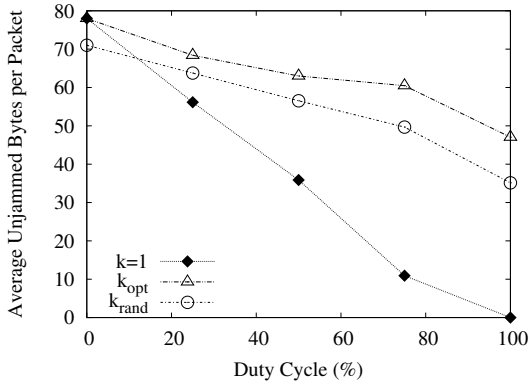


Fig. 5. Unjammed bytes per packet with duty-cycled jam-1 jammer

$\texttt{DATA}_{\texttt{unjammed}}$ for the senders and thus by forcing the jammer to transmit more jamming signals to disrupt an entire packet.

With no defense against the jammer, packets are extremely vulnerable to jamming. Even with short jamming signals such as jam-1, the jammer can disrupt $100\%$ of the packet (see Figure 4). However, when Jam-Buster is used, the receiver can successfully receive $44\%$ of the packet. Even when the jammer increases its jamming signals and uses jam-2, the receiver can still retrieve $16\%$ of the packet. By comparing $k = 1$ and $k_{rand}$ for no jammer, we can see that Jam-Buster achieves this high resilience by incurring only $9\%$ overhead due to additional CRCs. Essentially, this results in lower effectiveness of the jammer. With jam-1 and jam-2, the jammer now can only jam $56\%$ and $84\%$ of the packet indicating that if the jammer wants to disrupt the entire packet, the jammer needs to send even more jamming pulses per packet. Such high transmissions from the jammer eventually leads to a very fast jammer detection.

*Jam-Buster vs. Duty-cycled Jammers*

Our goal is to show that Jam-Buster exhausts the jammer's energy at a rate such that the jammer is either very short-lived or not effective at all. All the results in this section are for a jam-1 jammer.

Figure 5 shows the average unjammed bytes received while capturing the effect of the jammer's failure to detect packets

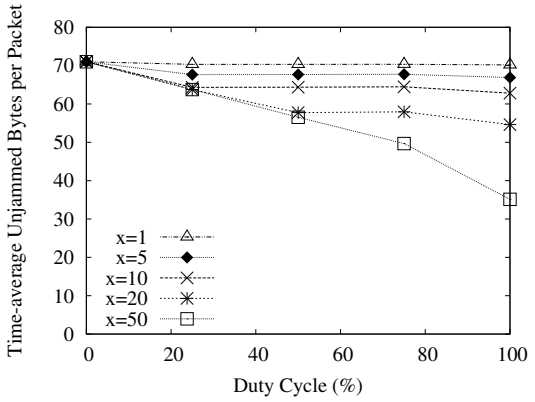


Fig. 6. Time-average unjammed bytes per packet for CBR-30

when it is asleep due to duty cycling. Essentially, this represents $\mathtt{d}_{jammer}.\mathtt{DATA}_{\mathtt{unjammed}}$ in the model. The jammer's failure to detect and jam packets due to duty cycling is obvious from the fact that the receiver successfully receives $72\%$, $46\%$ and $14\%$ of the packets when the jammer is following $75\%$, $50\%$ and $25\%$ duty cycles respectively. With `Jam-Buster`, the receiver receives even more data: $82\%$, $72\%$, $64\%$ and $45\%$ of the packet respectively. Essentially, this indicates `Jam-Buster`'s success in eliminating predictability.

Essentially, the energy exhaustion ratio, `life` depends on the number of senders, $n$ and CBR inter-arrival time, $T_d$. We observed a linear increase in `life` as the percentage of duty cycle increases since the jammer spends more time idle listening. The high idle listening cost of the jammer dominates the impact of the increased transmission cost at high values of $n$. As a result, we did not observe any change in `life` for the different node densities (Figure omitted due to space constraints). However, with lower traffic, we observed significant increase in `life`, a $275\%$ increase from CBR-5 to CBR-30. As traffic becomes low, the jammer spends more of its battery looking for packets than actually jamming the packets.

To evaluate `Jam-Buster`'s effectiveness, we considered jammer's effectiveness with different battery levels, $x$. Figure 6 shows the time-average unjammed bytes for CBR-30 for different battery levels of the jammer. For both the cases, we observed that, the $f_{RN}$s for the different duty cycles do not vary for $x = 1$. Nodes have the same time-average unjammed bytes. This is because of the relative lifetime. At low duty cycle, the jammer lives long but detects fewer packets during its lifetime. On the other hand, at high duty cycle, the jammer has shorter lifetime, but detects more packets while it is alive. When averaged out, the jammer jams the same number of bytes. As a overall effect, the receiver receives data as if there was no jammer in the system. As $x$ increases, the jammer lives longer, detecting more packets during its lifetime, thus reducing the time-average. However, the receiver still receives huge data. Even with a 20 times larger battery, the sender still can receive $71\%$ of the packet. With a jammer having $50$ times more energy, the jammer behaves like an always-on jammer. However, as the jammer transmits more, the sensors have higher probability of detecting the jammer. Hence, the jammer has a trade-off. For example, with 20 times battery, the jammer will do $50\%$ duty cycle as it begets the same benefit as an always-on jammer, but has slower detection.

## X. Conclusion

Current protocols for wireless sensor networks are extremely vulnerable to reactive jamming attacks because of the low resilience of the Zigbee packets, the easy differentiation of the packet types and the high predictability of the data transmission times. A smart jammer can exploit these points of vulnerabilities to pose energy-efficient jamming attacks that are hard to localize. We use a novel approach to defend against these cheap yet undetectable jammer. Instead of proposing anti-jamming solutions, we attack the jammer and force the jammer to transmit more jamming signals to achieve effective jamming. More transmissions from the jammer eventually results in faster detection of the jammer. Moreover, our solution, `Jam-Buster` forces the jammer to spend more energy effectively busting an energy-constraint jammer. Our experiments in the TmoteSky testbed show that `Jam-Buster` successfully reduces the effectiveness of the jammer and thus improves jam-resilience of the system.

## References


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

[2] A. Wood, J. Stankovic, and G. Zhou, "DEEJAM: Defeating energy-efficient jamming in IEEE 802.15. 4-based wireless networks," in *SECON*, 2007.

[3] W. Xu, W. Trappe, and Y. Zhang, "Channel surfing: defending wireless sensor networks from interference," in *IPSN*, 2007.

[4] "Boeing's 'virtual fence' on mexico border is halted," http://seattletimes.nwsource.com/html/nationworld/2011362012_border17.html.

[5] "Building a border: Part 2," http://www.denverpost.com/fortressamerica/ci_5356695.

[6] "Wildlife net-gamekeepers using sensor network," http://dl.acm.org/citation.cfm?id=1326269.

[7] "Students develop sensor network to monitor forest," http://www.informationweek.com/news/190301069.

[8] T. Rappaport, *Wireless communications*. Prentice Hall PTR, 2002.

[9] T. Instruments, "Cc2420 datasheet," *Reference SWRS041B, March*, 2008.

[10] D. MacKay, "Fountain codes," in *IEEE Communications*, 2005.

[11] S. Lin and D. Costello, *Error control coding*. Prentice-Hall Englewood Cliffs, NJ, 1983.

[12] V. Tarokh, N. Seshadri, and A. Calderbank, "Space-time codes for high data rate wireless communication: Performance criterion and code construction," *IEEE Transactions on Information Theory*, 1998.

[13] H. Huang, T. Huang, N. Chilamkurti, R. Cheng, and C. Shieh, "Adaptive forward error correction with cognitive technology mechanism for video streaming over wireless networks," in *3CA*, 2010.

[14] D. Thuente and M. Acharya, "Intelligent jamming in wireless networks with applications to 802.11 b and other networks," in *MILCOM*, 2006.

[15] Y. Law, L. van Hoesel, J. Doumen, P. Hartel, and P. Havinga, "Energy-efficient link-layer jamming attacks against wireless sensor network MAC protocols," in *Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks*. ACM, 2005, pp. 76–88.

[16] V. Paruchuri, S. Basavaraju, A. Durresi, R. Kannan, and S. Iyengar, "Random asynchronous wakeup protocol for sensor networks," 2004.

[17] L. Tang, Y. Sun, O. Gurewitz, and D. Johnson, "PW-MAC: An Energy-Efficient Predictive-Wakeup MAC Protocol for Wireless Sensor Networks," 2011.

[18] F. Ashraf, R. Crepaldi, and R. Kravets, "Know Your Neighborhood: A Strategy for Energy-efficient Communication," in *MASS*, 2010.

[19] A. El-Hoiydi and J. Decotignie, "WiseMAC: an ultra low power MAC protocol for the downlink of infrastructure wireless sensor networks," in *iscc*, 2004.