Energy-Efficient Link-Layer Jamming Attacks against Wireless Sensor Network MAC Protocols

Lodewijk van Hoesel Yee Wei Law[‡] Jeroen Doumen Pieter Hartel Paul Havinga
Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands
[‡]Riscure BV, Radex Innovation Center, Rotterdamseweg 183/C, 2629 HD Delft, The Netherlands
email: {lodewijk.vanhoesel, jeroen.doumen, pieter.hartel, paul.havinga}@utwente.nl, law@riscure.com

March 31, 2006

Abstract

A typical wireless sensor node has little protection against radio jamming. The situation becomes worse if energy-efficient jamming can be achieved by exploiting knowledge of the data link layer. Encrypting the packets may help to prevent the jammer from taking actions based on the content of the packets, but the temporal arrangement of the packets induced by the nature of the protocol might unravel patterns that the jammer can take advantage of, even when the packets are encrypted.

By looking at the packet interarrival times in three representative MAC protocols, S-MAC, LMAC and B-MAC, we derive several jamming attacks that allow the jammer to jam S-MAC, LMAC and B-MAC energy-efficiently. The jamming attacks are based on realistic assumptions. The algorithms are described in detail and simulated. The effectiveness and efficiency of the attacks are examined. In addition, we validate our simulation model by comparing its results with measurements obtained from implementation on sensor node prototypes. We show that it takes little effort to implement such effective jammers, making them a realistic treat.

Careful analysis of other protocols belonging to the respective categories of S-MAC, LMAC and B-MAC reveals that those protocols are, to some extent, also susceptible to our attacks. The result of this investigation provides new insights into the security considerations of MAC protocols.

General Terms: Security, algorithms.

Keywords: MAC protocols, security, denial-of-service attacks, jamming, clustering.

1 Introduction

Jamming-style DoS attacks on the physical and data link layer of wireless sensor networks (WSNs) have recently attracted attention [21, 43, 47, 48]. In particular, Xu et al. [47] propose 4 generic jammer models, namely (1) the constant jammer, (2) the deceptive jammer, (3) the random jammer and (4) the reactive jammer. A constant jammer emits a constant noise; a deceptive jammer either fabricates or replays valid signals on the channel incessantly; a random jammer sleeps for a random time and

jams for a random time; and lastly, a reactive jammer listens for activity on the channel, and in case of activity, immediately sends out a random signal to collide with the existing signal on the channel. According to Xu et al. [47], the constant jammers, deceptive jammers and reactive jammers are effective jammers in that they can cause the packet delivery ratio to fall to zero or almost zero, if they are placed within a suitable distance from the victims. However these jammers are also *energy-inefficient*, meaning they would exhaust their energy sooner than their victims would when given comparable energy budgets. Although random jammers save energy by sleeping, they are less effective.

Our contribution is to develop jamming attacks that (1) work on encrypted packets, (2) are as effective as constant/deceptive/reactive jamming, and (3) at the same time more energy-efficient than random jamming or reactive jamming. We implement such jamming attacks by exploiting the semantics of the data link layer and show the results quantitatively. The fact that our attacks are applicable to three representative MAC protocols suggests the same attacks are applicable to a wide range of other protocols belonging to the same categories as these protocols. Our analysis of the attacks provides new insights into the timing considerations of MAC protocols with regards to security, and provides hints on which category of protocols provides the best protection against our attacks so far, in the absence of effective countermeasures.

The motivation of this work stems from the concern that if an attacker can program and deploy a general-purpose link-layer jamming network that is able to jam any WSN effectively and energy-efficiently, and if a high entry barrier is not maintained for such a low cost attack, a WSN can never in any practical sense be secure. A counter-argument might be that energy efficiency is no concern to powerful attackers, but even powerful jammers come with a finite energy supply and they would advertise their presence and location if they simply blast away with an exorbitant amount of radio waves – this is something a sensible attacker would avoid.

The paper is organized as follows. We start by stating the assumptions on which our attacks are based in Section 2. We then describe the attack algorithms in Section 3. Section 4 describes how the protocols and the corresponding attacks are simulated, and how the results are evaluated. The results are given in Section 6. Implications of our work to other protocols are discussed in Section 7. Section 8 explores some potential countermeasures. In Section 9 we focus on one MAC protocol in particular; the LMAC protocol. We discuss the implementation of an efficient jammer for this protocol on prototype node hardware and we explore the effectiveness of LMAC countermeasure. Experiments are used to validate our simulation model for a small topology. Related work is discussed in Section 10. Finally Section 11 concludes.

2 Assumptions

We assume an attacker has two goals: the primary goal is to disrupt the network by preventing messages from arriving at the sink node, and the secondary goal is to increase the energy wastage of the sensors. A sink node is a node that requests, and hence sinks, information. Our attacks depend on three assumptions: (1) the jammer nodes know the preamble[1] used by the victim nodes, (2) the jammer nodes can measure the length of a packet, and (3) the jammer nodes know what MAC protocol the victim nodes are running.

[1] A preamble is a bit sequence, usually consisting of alternating 1's and 0's, for training the receiver, and its length depends on the data coding scheme used [36].

Requirements (1) and (2) should be easy to satisfy in practice. Requirement (3) is more demanding, but not impractical to satisfy – as our future work, a strategy will be devised to map observed traffic to specific classes of protocols. Note that the jammer motes do not need to know the content of the packets, so our attacks work even if the packets are encrypted. Adding to the significance of our attacks is that the attacker does not need to capture and compromise any existing sensor nodes.

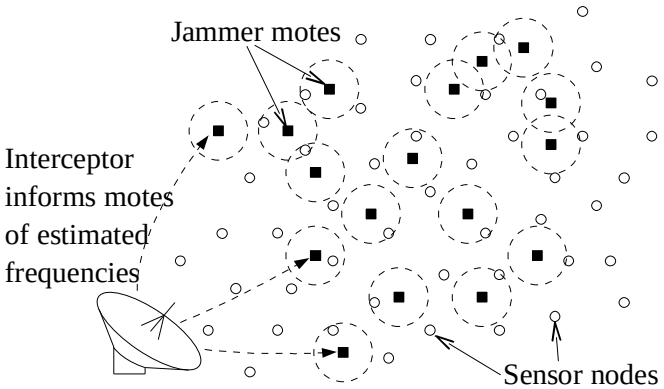


Figure 1: Distributed jamming.

We now describe the concrete set of circumstances under which our attack scenario is applicable.

- There are no fixed sink nodes to attack, e.g. in directed diffusion [13] where ID-less interests propagate through the network, there is no way for a node to tell where the real sinks are.
- It is infeasible for the attacker to deploy its nodes strategically, except perhaps to air-drop them on top of the target network.
- The attacker can only estimate where and not how the target network is or would be deployed.
- The target network is protected with a link-layer authentication (and optionally encryption) scheme like TinySec [16].

In all these cases, the attacker would find it appealing to distribute its jammer nodes among the target WSN and apply our jamming attacks. We are in fact not alone in suggesting this attack scenario [29]. One note of caution though, is that if the target network employs frequency-hopping spread spectrum [1], the attacker would find it necessary to deploy an interceptor to discover the hopping frequencies first before its network of jammer nodes can start jamming (Figure 1).

Naturally, our attacks are affected by the choice of values assigned to the protocol parameters. Throughout the paper, we pick values for the protocol parameters that are either as realistic or as faithful to what the creators of the protocols recommend as possible, in the absence of a universal consensus and a scientifically rigorous way of deriving these values.

3 MAC Protocols for WSNs

Before we describe our attacks, a brief overview of MAC protocols for WSNs is in order. Despite the number of protocols proposed so far, there is still no clear indication of the mechanisms the proposals are converging to [20]. However, since different applications require optimizing different parameters, there will most likely not be a single solution, that fits all types of applications. According to Langendoen et al.'s survey [20], WSN MAC protocols can be classified according to (1) the number of channels used, (2) how the intended receiver of a message is notified, and (3) how medium accesses are temporally organized.

In terms of channels, most protocols use only a single channel, e.g. S-MAC [49, 50], LMAC [40] and B-MAC [31].

In terms of message notification, in some protocols, a communication scheduling algorithm determines when a node listens for messages to minimize energy consumed by idle listening. This type of protocols is typically either more resource-demanding or requires architectural support [18, 22, 32]. In other protocols, a node has to determine on its own when to listen for messages. To reduce energy spent on idle listening, they typically employ some form of sleep-listen schedule. Since these protocols are more lightweight and hence more viable for current WSNs, we concentrate on this type of protocols, of which S-MAC, LMAC and B-MAC are, again, well-known and widely used examples.

In terms of temporal organization of medium accesses, S-MAC, LMAC and B-MAC belong to different categories. S-MAC divides time into slots, and nodes contend for slots to send packets. LMAC divides time into frames, and each node is allocated a slot in the frame to send packets. B-MAC uses random accesses to the communication medium, i.e. no slots and no frames, to send packets.

Based on the above analysis, S-MAC, LMAC and B-MAC are representative of current WSN MAC protocols, and our jamming analysis in the following will hence be targeted at these protocols. What follows is a brief introduction to each of the protocols.

3.1 S-MAC

The design of S-MAC revolves around a periodic sleep-listen schedule. A period is divided into a listen interval and a sleep interval. The listen interval in turn consists of a SYNC interval and a CTRL interval. The sleep interval allows the nodes to sleep in order to reduce the amount of energy spent on idle listening. The ratio of the length of the listen interval to the length of the period is the duty cycle. Lowering the duty cycle based on a fixed period reduces energy usage.

We now describe the operation of S-MAC. When a node A first joins a network, it listens for a whole period. If the channel is clear, A broadcasts its schedule in a SYNC packet, telling its neighbors that it will sleep t_s seconds later, marking the end of A's listen interval. If a node B receives A's schedule before choosing its own, B will adopt A's schedule and after a random delay of t_d seconds, broadcast to tell A and other neighbors that it will sleep at $t_s - t_d$ seconds later. Should B receive A's schedule after broadcasting its own, it adopts both schedules [50]. In this way, A, B and their neighbors are able to synchronize their schedules.

Data packets are mainly sent during the CTRL interval, and may extend into the sleep interval. When broadcast, data packets are sent without the exchange of RTS/CTS packets (RTS = Request-To-Send, CTS =Clear-To-Send). When unicast, RTS/CTS

packets are exchanged. Collision avoidance depends solely on carrier sense and the use of network allocation vectors [50].

3.2 LMAC

LMAC is a TDMA protocol. In LMAC, time is divided into frames, which are further divided into time slots. In each frame, a sensor node takes control of one time slot (or more, for instance in a variant of LMAC called AI-LMAC [7]). A time slot is further divided into two parts of unequal length: (1) a control part for transmitting a control packet, and (2) a data part for transmitting a data packet. In the time slot it controls, a node always starts by sending out a control packet even if it does not have any data to send. Besides addressing other nodes, the control packet is also necessary for maintaining synchronization. When the neighbors of the node discover by listening to the control packet that they are not the intended receivers, or that the node simply has no data to send, they immediately turn off their receivers and sleep until the next slot.

The neighbor addressed by the control packet stays listening. The data packet is transmitted right after the control packet. The absence of RTS/CTS signalling makes LMAC a particularly energy-efficient protocol, however tight time synchronization is required.

3.3 B-MAC

The central feature of B-MAC is its preamble sampling scheme, called low power listening (LPL), which is a continuation of a tradition set by El-Hoiydi [9], and Hill and Culler [11]. As one of the main sources of energy wastage in WSNs is idle listening, a simple solution is to listen and sleep periodically according to some duty cycle. The requirement for monitoring-type applications [24] to reduce the duty cycle to 1% (i.e. listen for 1% of an entire cycle) means that the sensor nodes should listen for the briefest time possible. Preamble sampling achieves this by delegating to the transmitter the responsibility of making sure the receiver receives the packet, in that the transmitter must transmit a preamble that is long enough to be sensed by the receiver which only wakes up for the briefest moment and sleeps most of the time (Figure 2). Note that this is different from S-MAC in that the sender and receiver are not synchronized.

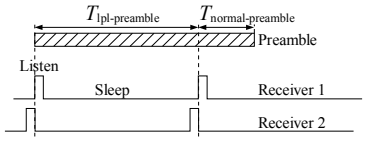


Figure 2: Preambles should be long enough such that a receiver listening at the very beginning of the cycle, or at the very end of the cycle would be able to detect the preamble.

Unlike S-MAC or LMAC, B-MAC is only a link protocol, in that it does not stipulate how the communication medium is shared between nodes. However, it is reasonable to assume RTS/CTS signalling is used. When RTS/CTS signalling is used, the sender sends an RTS packet with an LPL preamble. Upon detecting this long preamble, the receiver snaps out of the LPL mode, and replies with a CTS packet that has a normal preamble, since the sender is already listening and waiting. The ensuing data

packet and acknowledgement packet exchanged between the sender and receiver are all transmitted with a normal preamble. After sending the acknowledgement packet, the receiver returns to the LPL mode.

This concludes our summary of the most widely used MAC protocols in WSNs.

4 Description of Attacks

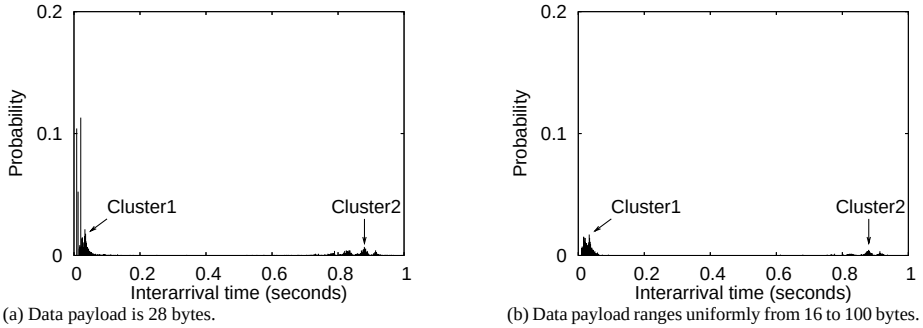
Imagine we are the attacker; the question now is how do we attack a protocol without knowing the content of the packets? Since the attacker is solely interested in jamming data packets, our first observation is that since data packets are longer than control packets, we can focus on jamming long packets. We can do this by sorting packets according to their length and predict when long packets would arrive. This strategy might not work however because (1) data packets might be generated spontaneously, rendering our prediction inaccurate, and (2) data packets are sparse, e.g. 1 packet every 5 minutes from each node [24]. Sparse packets require us to observe for a long time before we get a working prediction model, and offer us little opportunities to re-adjust our prediction.

A more promising approach is to look at the probability distribution of the interarrival times between packets, i.e. packets of all types. We look at S-MAC, LMAC and B-MAC in turn.

4.1 S-MAC

For example, Figure 3 shows the probability distribution of the packet interarrival times observed by a node, which has 6 neighbors that send data to a sink multiple hops away every 5 seconds, in a static network using S-MAC with a period of 930 ms and a duty cycle of 10% (i.e. default values that come with the original S-MAC source code). There are 2 clusters in the graph. Let us call them Cluster1 and Cluster2. Although using different data packet lengths results in different shapes of the clusters (e.g. distinct spikes in Cluster1 in Figure 3(a) in contrast to Figure 3(b)), the clear separation between the clusters still stands. These two clusters can also be observed even if the nodes are mobile.

These clusters are *not* due to the periodic nature of data reporting by the nodes, in fact they are solely the result of the periodic nature of the protocol, in this case S-MAC itself. The explanation is as follows. In S-MAC, packets within a period are closely spaced in time, accounting for the interarrival times in Cluster1. Two packets from two different periods are more widely spaced because of the sleep interval, and the interarrival time between these two packets falls in Cluster2. Unless the nodes insist on sending only 1 packet every period, which is improbable, it is only natural that Cluster1 has a larger weight, or higher probability, than Cluster2. Observe that Cluster2 has a larger variance than Cluster1. One way to understand this is to compare two Cluster2 interarrival times: the time separation between two SYNC packets of two consecutive periods is large, but the time separation between an acknowledgement packet and a SYNC packet of the *subsequent* period is small, and yet these two time separations belong to Cluster2. Actually, there are other clusters at the further end of the time axis, but their probability is negligible. They have to be filtered out in order for clustering to work. It is reasonable, for example, to expect S-MAC to have a *maximum* period of 1 s (otherwise the latency would be large), thus filtering all interarrival times larger than 1.5 s should eliminate these unwanted clusters.



(a) Data payload is 28 bytes.

(b) Data payload ranges uniformly from 16 to 100 bytes.

Figure 3: Probability distribution of packet interarrival times for S-MAC with a period of 930 ms and a duty cycle of 10%. Choices of packet lengths and reporting frequency are detailed in Section 5.

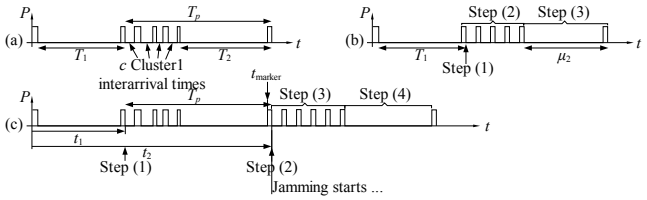


Figure 4: Jamming strategy for S-MAC: (a) actual timing, (b) naive version, (c) periodic clustering-based jamming (PCJ).

Our S-MAC attack strategy follows from the following deduction. If according to observation, for every Cluster2 interarrival time, there are c Cluster1 interarrival times, then right after observing a Cluster2 interarrival time, we should expect around c Cluster1 interarrival times (Figure 4(a)). Denote the means of Cluster1 and Cluster2 interarrival times as μ_1 and μ_2 respectively. A straightforward strategy is to first collect a reasonable number (e.g. 64) of consecutive interarrival times, and then perform clustering on them. The result of clustering gives us μ_1 and μ_2. The next steps are then to (Figure 4(b)):

1. Wait until a single Cluster2 interarrival time, T_1, is observed.
2. Jam with c packets, with a space of μ_1 seconds in between.
3. Sleep for μ_2 seconds.
4. Repeat the cycle starting from step 2.

This strategy however does not work in practice because μ_2 is often not a good approximation of T_2 due to the large variance of Cluster2. An improvement is to estimate T_p instead. Since T_p is a sum of several Cluster1 interarrival times and T_2, and since the variance of Cluster1 is smaller, T_p can be predicted more accurately. Our improved strategy is thus to (Figure 4(c)):

1. Wait until a single Cluster2 interarrival time is observed and record arrival time as t_1.

2. Wait for another Cluster2 interarrival time and record arrival time as t_2. Calculate the period $T_p = t_2 - t_1$. Record length of packet in time (not in bytes, as with all packet lengths that appear hereafter) just received as L_p.

3. Set t_{marker} = current time - L_p. Jam with $c - 1$ packets, with a space of μ_1 seconds in between. (Notice where this step starts in Figure 4(c).)

4. Sleep until $t_{\text{marker}} + T_p$.

5. Set t_{marker} = current time. Jam with c packets, with a space of μ_1 seconds in between.

6. Repeat the cycle starting from step 4.

Periodic re-estimation is done by repeating the cycle starting from step (1) instead of step (4). We call this approach *periodic clustering-based jamming* (PCJ), and depending on the context we also use PCJ to mean a periodic clustering-based *jammer*.

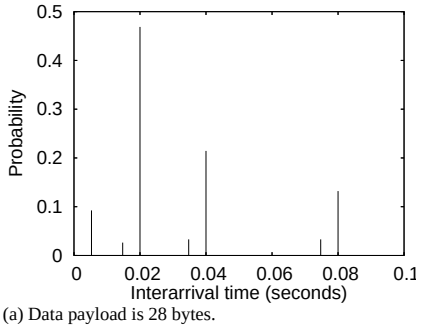
Additionally, we propose RPCJ, a reactive version of PCJ, by modifying step (3) of PCJ. In step (3) of RPCJ, the jammer records current time t_{marker} as before, but instead of jamming proactively, the jammer listens for a preamble by setting a timer that expires $\max(T_{\text{Cluster1}})$ seconds later, where $\max(T_{\text{Cluster1}})$ is the maximum interarrival time in Cluster1. If a preamble is detected, it jams the ensuing frame, otherwise if the timer expires, that is, if no preamble is detected, the jammer sleeps until $t_{\text{marker}} + T_p$ seconds, before waking up to listen for preambles again. Note that the jammer transmits only random packets, without any preamble.

The above attack relies on data clustering. A simple algorithm like K-means [25] is sufficient, because of the clear separation between the clusters, and the distinctness of each of the clusters. K-means involves only simple multiplications. A 32-bit floating-point hardware-accelerated multiplication (or division) consumes only 9 cycles on a typical sensor node CPU like the MSP430F149 [38]. A K-means iteration consists of an assignment step and an update step [25]. Denote the number of clusters as K and the number of data samples as N. The assignment step takes KN multiplications, whereas the update step takes KN multiplications and K divisions (having the same cost as multiplications). Assuming multiplication is the most expensive operation, then the computational complexity of a K-means iteration is $K(2N+1) \approx 4N$ multiplications, taking K as 2. According to simulations, only 2 iterations are usually required, 3 and above are *rare*. So for example, if $N = 64$, the energy required for 2 iterations is at least 4.4 μJ, or 60% of the energy required to transmit one bit (7.4 μJ) on a CC1000 radio [45] (both MSP430F149 and CC1000 are common components in existing sensor nodes). From the jammer's perspective, the computational cost is likely to be justified, since the jammer has nothing else to do apart from jamming.

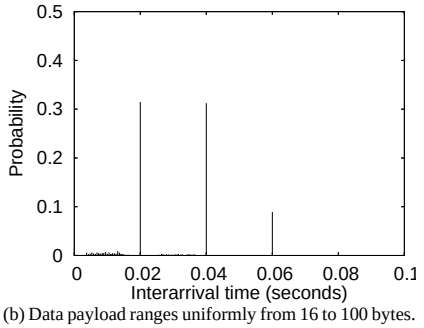
4.2 LMAC

Figure 5 shows clusters that are clearly spaced out at integral multiples of the slot size – typical of a TDMA protocol. However, there might be one or more clusters before the cluster centered on the slot size, depending on the distribution of the data packet length. Unlike S-MAC, the clusters at the higher end of the time scale are not negligible, so the jamming algorithm suggested for S-MAC cannot be applied.

The jammer's objective is to estimate the slot size by calculating the mean of T_S (Figure 6), μ_s, and to jam the beginning of every slot. The algorithm is based on two assumptions. **The first assumption** is that T_S can indeed be observed, i.e. the



(a) Data payload is 28 bytes.



(b) Data payload ranges uniformly from 16 to 100 bytes.

Figure 5: Probability distribution of packet interarrival times for LMAC with a slot size of 20 ms and 20 slots in a frame.

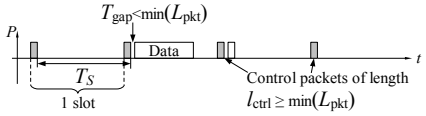


Figure 6: Interarrival times in LMAC.

probability that at least two occupied slots are consecutive is at least larger than 0.5. Using elementary combinatorics, it can be shown (in the Appendix) that

$$\Pr\{\text{at least two occupied slots are consecutive}\}$$

$$= \begin{cases} 0 & \text{if } 0 \leq n < 2 \\ 1 - \frac{\binom{s-n}{n}}{\binom{s-1}{n}} & \text{if } 2 \leq n \leq \frac{s-1}{2} \\ 1 - \frac{2\binom{s-n-1}{n-1}}{\binom{s}{n}} & \text{if } n = \frac{s}{2} \ (s \text{ is even}) \\ 1 & \text{if } \frac{s}{2} < n \leq s \end{cases} \tag{1}$$

In Equation 1, s ($s \geq 4$) is the total number of slots in a frame, and n ($0 \leq n \leq s$) is the number of occupied slots in a frame. When s is even and $n = \frac{s}{2}$, the probability is always larger than 0.5. In other words, for a given s, a node is more likely to observe at least two consecutive occupied slots, if n, the number of occupied slots, at least satisfies Equation 2:

$$\frac{\binom{s-n}{n}}{\binom{s-1}{n}} = \prod_{j=1}^{n-1} \left(1 - \frac{n}{s-j}\right) < 0.5 \tag{2}$$

For example, if $s = 20$, the least n that satisfies Equation 2 is $n = 4$. Given that most practical WSNs are dense [3], this requirement is almost certainly satisfied.

The second assumption is that the shortest data packet *might* be shorter than a control packet, but the longest data packet *must* be longer than a control packet, i.e.

$$\min(L_{\text{pkt}}) \leq l_{\text{ctrl}} < \max(L_{\text{pkt}}) \tag{3}$$

Here L_{pkt} is the random variable representing packet lengths (regardless of packet type), and l_{ctrl} is the fixed length of a control packet. For example, a control packet

takes 15 bytes in an optimized implementation (or 23 bytes in our unoptimized implementation), a data packet header takes 7 bytes, a message authentication code takes 4 bytes, so if a data packet payload is less than $15 - 7 - 4 = 4$ bytes (or 12 bytes in our non-optimized implementation), the corresponding data packet would be shorter than a control packet. The control packets in LMAC are longer than those in S-MAC because they contain more information like the slot occupancy vector, the number of hops to the gateway, etc. This assumption unfortunately bars us from estimating T_S by just measuring the interarrival time between two neighboring shortest packets, since we can no longer be sure if the two neighboring shortest packets are both control packets.

The algorithm itself consists of the following steps:

1. Suppose the observed packets are P_0, P_1, ... Denote the interarrival time between packet P_i and packet P_{i+1} as t_i, and the length of packet P_i as l_i ($i = 1, 2, \ldots$). Store t_1, t_2, ... in the ordered set $\mathcal{T}$, and l_1, l_2, ... in the ordered set $\mathcal{L}$. Had there only been data packets and no control packets, the jammer's job would have been easier, since the slot size is then simply

$$\mu_S = t_i - l_{i+1} + l_i \tag{4}$$

   But there are control packets, so the jammer has to continue as follows.

2. This step is based on two observations. The first observation is that T_S has to be large enough to accommodate both a control packet and a data packet, i.e.

$$T_S > L_{\text{ctrl}} + \max(L_{\text{pkt}}) \geq \min(L_{\text{pkt}}) + \max(L_{\text{pkt}}) \tag{5}$$

   The last inequality is the result of Equation 3. Equation 5 gives a lower bound for the slot size. The second observation is that a slot can accommodate at most 2 packets, so it is always smaller than the sum of 3 contiguous interarrival times, i.e.

$$T_S < \min(t_i + t_{i+1} + t_{i+2}) \qquad \text{where } 0 \leq i \leq |\mathcal{T}| - 2 \tag{6}$$

   Equation 6 gives an upper bound for the slot size. Denote the lower bound and the upper bound respectively as a_0 and a_m. Set $a_0 = \min(\mathcal{L}) + \max(\mathcal{L})$ and $a_m = \min(t_i + t_{i+1} + t_{i+2})$.

3. Compute the probability mass function of the interarrival times in the interval $[a_0, a_m)$. For this purpose, we can quantize the time interval $[a_0, a_m)$ into m millisecond-strips, $[a_0, a_1)$, ..., $[a_{m-1}, a_m)$, as practical values for the slot size are in milliseconds, and count the number of interarrival times that fall within each strip. Denote the strip with the highest count, i.e. the highest probability mass, as $[a_i, a_{i+1})$ ($0 \leq i < m$).

4. If a higher accuracy is desired, set $a_0 = a_i$ and $a_m = a_{i+1}$ and repeat the previous step with a finer time resolution. If not, set μ_S, the estimated slot size, as the mean of the interarrival times that lie in $[a_i, a_{i+1})$. If t_i is the time closest to μ_S, set μ_L, the estimated control packet length, as l_i. Cautionary note: this estimate might be wrong if the probability given by Equation 1 is not high enough. For example, if in Figure 5, the probability density at 40 ms (twice the slot size) is higher than the probability density at 20 ms (the slot size), i.e. two occupied slots are more likely to be separated by an unoccupied slot than to be consecutive, the estimate is double the real slot size. However, this tends to happen only at the fringe of the network, where the network density is lower. Furthermore, if

two occupied slots are indeed more likely to be separated than consecutive, the jammer would not miss much by jamming every two slots instead of every slot.

5. Listen for a packet that is of size μ_L. Once received, transmit a jamming packet, and sleep until time = current time - μ_L + μ_S. It is true that a packet of size μ_L might or might not be a control packet, in view of Equation 3. If the received packet is indeed a control packet, then the jammer is able to synchronize neatly with the LMAC schedule, allowing it to jam the control packet of every slot. If the received packet is not a control packet however, the jammer's schedule is offset by at least $L_{\text{ctrl}} + \mu_L$, but it is still able to jam the data packet, if there is any, of every slot.

6. Wake up, transmit a jamming packet, and sleep until μ_S seconds later. Repeat this step until when periodic re-estimation is required, in which case go to step (1).

We call this algorithm *periodic slot-based jamming* (PSJ). In the reactive version of the algorithm (RPSJ), the jammer listens for a preamble before jamming, instead of jamming proactively.

4.3 B-MAC

The probability distribution of packet interarrival times of B-MAC does show some clusters but they cannot be taken advantage of because B-MAC uses a periodic cycle only for *listening*, and not sending – we cannot periodically jam something that is not periodically sent. However, it is exactly this periodic listening that the jammer can take advantage of to save energy. Since a B-MAC has to listen every, say 10 ms, for a valid preamble, the jammer can be sure that if it samples the RSSI every 10 ms, it would be able to hear whatever preamble is being sent. The jamming strategy is hence to find out the check interval the victim nodes are using. This can be achieved by finding out the length of the longest observed preamble. Assuming this observed LPL preamble is $T_{\text{lpl-preamble}}$ seconds long, and the length of the normal preamble is $T_{\text{normal-preamble}}$, according to Figure 2, the jammer should sample the channel every ($T_{\text{lpl-preamble}} - T_{\text{normal-preamble}}$) seconds. The jammer can either guess a value for $T_{\text{normal-preamble}}$, or listen to the channel for the shortest preamble used. If the jammer chooses to guess, to be on the safe side, the jammer should choose a value that is slightly larger than the typical value, which is three to four bytes [36], so that the jammer would sample the channel slightly more frequently than the victim nodes do. We call this approach *LPL-based jamming* (LPLJ).

In the next two sections, we will explain how the attacks are simulated before presenting the results. Then we will explore some potential countermeasures.

5 Simulation and Evaluation Model

All protocols, attacks and countermeasures are simulated using the OMNeT++ framework (`www.omnetpp.org`). A simulation consists of a sink node, $(N_r - 1)$ router nodes, N_s source nodes and N_j jammer motes, all capable of a radio range of r, and located in a square $l \times l$ area.

The sink is positioned in the middle of the area, whereas the router nodes are pseudo randomly placed at most r from the sink. Both the source nodes and the jammer motes

are pseudo randomly placed more than r away – this is to avoid the jammer motes having direct effect on the sink, giving the jammer motes an unfair advantage and to allow us to investigate the effect of jamming on the routing of information from the sources to the sink. We require the *node density* to be uniform across the simulation area, i.e. $\frac{1+N_r-1}{\pi r^2} = \frac{N_r+N_s}{l^2}$, or $l = r\sqrt{\left(1+\frac{N_s}{N_r}\right)\pi}$, to avoid the peculiarities of any specific non-uniform topology having an effect on jamming. In the absence of jammer motes, the *network density* [51] is then $D = \frac{(1+N_r-1+N_S)\pi r^2}{l^2} = N_r$, i.e. equivalent to the number of sink and router nodes. To simulate attacks, the jammer motes are activated 10 seconds after the sensor network starts operating, to allow the sensor nodes to finish discovering their neighbors and settle down into a steady state before the jamming starts, thereby simulating the attack scenario described in Section 2. The total simulation time is T_{sim} virtual seconds. Every experiment is run I times, with different seeds each time. The network topologies are simulated as static. The values of the parameters are summarized in Table 1 and are chosen to satisfy the constraints of memory and time available for simulations (maximum 900 MB of RAM and 2 actual minutes per run).

Table 1: Simulation parameters and notations.

MAC	N_s	T_{sim}	I
S-MAC	40	600	10
LMAC	20	200	10
B-MAC	20	200	10

N_s = number of source sensor nodes
T_{sim} = simulated time in seconds
I = number of runs
N_j = number of jammer motes (set to $0.75N_s$ or N_s)
D = network density (set to 15 or 20)
N_r = number of sink and one-hop neighbors of the sink

On the application layer, the sink node broadcasts an interest once it found a neighbor but it only broadcasts the interest once throughout the simulation. The source nodes each broadcast a matching data every 5 seconds, as an approximation of a network with moderately fast-changing data [7]. The data packet payload ranges uniformly from 16 bytes to 100 bytes. The minimum payload corresponds to a TinyDiffision [26] payload of 2 attributes (the least number of attributes). The maximum payload is a popular choice [31, 39], and it corresponds to a TinyDiffusion payload of 23 attributes. While a uniform distribution of packet sizes is not realistic, it serves as a 'base case' that allows us to investigate the capability of jammers in reaction to a wide range of packet lengths.

On the network layer, TinyDiffusion [26], faithfully ported from TinyOS (`tinyos.sf.net`), is used for S-MAC and B-MAC. LMAC has a simple built-in routing algorithm, and so LMAC interfaces directly with the application layer.

On the data link layer, S-MAC is simulated with *adaptive listening* [50], with a period of 930 ms and a duty cycle of 10%. The code is also faithfully ported from TinyOS. LMAC is simulated with a fixed slot size of 20 ms (a little more than enough to fit 100-byte payloads) and 20 time slots per frame (suitable for a network density of 20), using the same codebase from our previous work [40]. The B-MAC code is built on top of the LPL code from TU Delft's MAC simulator [20]. Following Polastre et al.'s choice [31], we use an RSSI sampling time of 350 μs and a check interval of 100 ms for B-MAC. We also implement RTS/CTS signalling on top of the core B MAC protocol. Both S-MAC and B-MAC use a contention time of 41 ms (the default given by the original S-MAC source code).

On the physical layer, the radio characteristics follow those of the RFM TR1001 [35]. The ratio between the power consumption in sleep, Rx and Tx mode is 1:960:2400.

Switching times are taken into account. 8-to-12 bit data encoding scheme [34] is assumed, so an encoded data is 1.5 times the size of the original raw data. Denote T_{preamble} as the time required to transmit a preamble, T_{byte} the time required to transmit one byte and L_{pkt} the number of bytes in a packet on the data link layer. The total length of a frame (i.e. packet on the physical layer) in time is then given by

$$T_{\text{frame}} = T_{\text{preamble}} + (1 + 1.5L_{\text{pkt}})T_{\text{byte}} \tag{7}$$

The extra 1 byte in the parenthesis is to account for the start byte, which is used to align the incoming bit stream. In the simulation, $T_{\text{preamble}} = 5T_{\text{byte}}$ and $T_{\text{byte}} = 10/115200$. The extra 2 bits in T_{byte} is to account for 1 start bit and 1 stop bit at the beginning and at the end of a byte.

To simulate jamming, the jammer emits a random packet that is *at least* T_{preamble} seconds long if the jamming starts from the start of the preamble. If the jamming starts from the end of the preamble as is the case with all types of reactive jamming, the jamming packet is only T_{byte} seconds long. It is assumed that the integrity of a packet is protected by a message authentication code, and a corrupted bit is enough to nullify the validity of the packet, therefore corrupting one byte, or 8 bits (i.e. the minimum that can be transmitted conveniently), should be sufficient to corrupt the whole packet. The width of the jamming pulse has a large impact on the simulation outcomes.

Some aspects of the jammers are as follows. The random jammer is simulated to sleep, and jam for a time uniformly distributed between 1 ms and 500 ms. There are several tuneable parameters for PCJ, RPCJ, PSJ and RPSJ. For example, all PCJ and RPCJ implementations start with a minimum sample size of 64 interarrival times, and readjust their estimation every 8 periods. Tuning these parameters allows the attacker to dynamically adjust its behavior, but the effects of such tunings are left to future investigation. The fact that jammer motes have limited buffer for storing interarrival times and packet lengths is realistically reflected in the implementations. The clustering operations are directly translatable to hardware implementations.

Among the things that are not simulated are (1) processing delays, (2) interference and (3) the gray area effect [52]. We will discuss processing delays in relation to our results later. As for interference, ideally the SNR model of Reijer et al. [33] should be used. However since the model takes into consideration all nodes in the network other than the sender for *every* transmission, it demands far greater computational resources than what is required of the conventional, circular model that only considers all nodes in the radio range of the sender. To see how we can implement the more accurate SNR model efficiently is our future work. As for the gray area effect, so far there is still no comprehensive theoretical model for simulating it. All in all, these are some of the many simplifications that are conventionally applied in simulations, as is the case that simulated radio ranges are circular/spherical by convention –a departure from actual measurements [33].

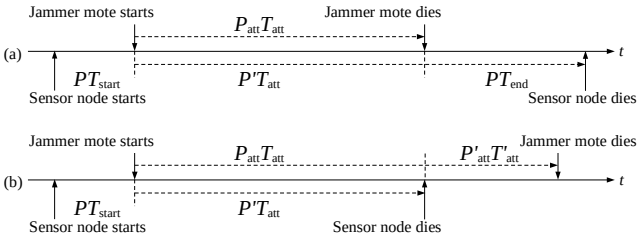
5.1 Metrics

Following our previous work [21], we evaluate how *effective* an attack is by measuring primarily the *censorship rate* R_c and secondarily the *attrition rate* R_a. R_c measures the fraction of messages blocked. Let M be the number of messages arriving at the sink in the absence of attacks, and M' be the number of messages arriving at the sink in the presence of attacks, then

$$R_c = (M - M')/M \tag{8}$$

R_a measures the fraction of additional energy the sensor network has to spend in the presence of attacks. Let E be the amount of energy spent when there is no attack, and E' be the amount of energy spent when there is attack, then

$$R_a = (E' - E)/E \tag{9}$$



P	=	power consumed by a sensor node when not under attack
P'	=	power consumed by a sensor node when under attack
P_{att}	=	power consumed by a jammer mote when attacking
P'_{att}	=	power consumed by a jammer mote when not attacking
T_{att}	=	duration of a jammer mote's attack
T'_{att}	=	remaining lifetime a jammer mote after finishing attack
T_{start}	=	time between when the sensor node starts operating and when jamming begins
T_{end}	=	time between when jamming ends and when sensor node dies

Figure 7: Calculation of lifetime advantage: (a) if the jammer mote dies before the sensor node, or (b) otherwise.

To evaluate how *energy-efficient* an attack is, we use the *effort ratio* R_e, defined as the ratio of the attacker's per-node energy expenditure to the sensor network's per-node energy expenditure when not under attack. We can compare the effort ratio of a random jammer and the effort ratio of a reactive jammer as follows. Given the same target protocol, if the power consumption ratio between the sleep, Rx and Tx mode is $1 : \rho : \tau$, a reactive jammer that jams j ($j < 1$) of the time has a *higher* effort ratio than a random jammer only when Equation 10 is satisfied:

$$j > \frac{1}{2}\left(1 + \frac{1-\rho}{\tau-\rho}\right) \tag{10}$$

Plugging in the values $\rho = 960$ and $\tau = 2400$ as given in the previous section 5, we get $j > 17\%$.

Using R_a and R_e, we can calculate the *lifetime advantage* R_l of a jammer node over a sensor node, that is how long a jammer node can live compared to a sensor node. To derive an expression for R_l, the following assumptions are used: (1) a jammer node has the same total amount of energy as a sensor node, and (2) the power usage is constant. There are two scenarios. Equation 11 is for the scenario where the jammer mote dies first (Figure 7(a)).

$$R_l = \frac{T_{\text{att}}}{T_{\text{att}} + T_{\text{start}} + T_{\text{end}}} = \frac{P}{P - P' + P_{\text{att}}} = \frac{1}{R_e - R_a} \tag{11}$$

Equation 12 is for the scenario where the sensor node dies first (Figure 7(b)).

$$\begin{aligned} R_l &= \frac{T_{\text{att}} + T'_{\text{att}}}{T_{\text{start}} + T_{\text{att}}} = \frac{P}{P'_{\text{att}}} + \left(1 + \frac{P' - P_{\text{att}} - P}{P'_{\text{att}}}\right)\frac{T_{\text{att}}}{T_{\text{start}} + T_{\text{att}}} \\ &\approx 1 + \frac{P' - P_{\text{att}}}{P'_{\text{att}}} \geq 1 + \frac{P' - P_{\text{att}}}{P_{\text{att}}} = \frac{1 + R_a}{R_e} \end{aligned} \tag{12}$$

In Equation 12, the $\approx$ relation is because $T_{\text{start}} \ll T_{\text{att}}$; the $\geq$ relation is because $P'_{\text{att}} \leq P_{\text{att}}$ for a jammer mote that has nothing more to jam, consistent with our assumptions of the attack model. Equation 12 gives only the lower bound but in cases where the jammer mote outlives the sensor node, knowing the lower bound is good enough.

6 Simulation Results

The censorship rates and lifetime advantages of the various attacks are given in Table 2. For readability's sake, the figures for attrition rate and effort ratio are omitted. We start by making some general observations on the results. First, the censorship rates, both the means and the standard deviations, improve with N_j/N_s and D (defined in Table 1). This is in line with intuition: more jammer motes have greater jamming effect, and the more sensor nodes a jammer mote has as neighbors, the sooner the jammer mote can synchronize with the S-MAC/LMAC schedule, or determine the preamble length in B-MAC's case. Note that in all simulated cases, $N_j < N_s + N_r = N_s + D$. Therefore further increase in censorship rate can be expected if we increase the number of jammer motes, N_j, to the total number of sensor nodes, $N_s + N_r$.

The second general observation is that our jamming algorithms outperform random jamming and reactive jamming in lifetime advantage as expected since random jamming is not a targeted effort, and reactive jamming consumes energy constantly in listening.

Thirdly, although it appears in Table 2 that random jamming is not as effective against LMAC and B-MAC as it is against S-MAC, the results are to be interpreted with caution. In LMAC's case for example, the jammer's random sleep interval, uniformly distributed between 1 ms and 500 ms (25 times the slot size), is most of the time wide enough to allow many data packets to pass through. By reducing the jammer's random sleep interval, we can get the censorship rates equally close to 100% for all S-MAC, LMAC and B-MAC. We do not customize the random sleep interval for each of the protocol in our simulations however, because as the random sleep interval gets smaller, the jammer consumes more energy switching between the sleep mode and the transmission mode – this switching energy becomes a dominant component of the overall energy consumption, increasing the jammer's effort ratio. This is to say that although we know higher censorship rates can be achieved by customizing the random jam/sleep interval according to the target protocol, we do not, because the result only makes random jamming more effective than it already is now and not any more efficient.

Finishing our general observations, we now analyze the results in more detail below.

6.1 S-MAC

We compare random jamming, reactive jamming with the following energy-efficient attacks: listen interval jamming (LIJ), control interval jamming (CIJ), data packet jamming (DPJ), periodic cluster-based jamming (PCJ) and reactive periodic cluster-based jamming (RPCJ). LIJ, CIJ and DPJ are three jamming algorithms we introduce in our previous work [21] that work on unencrypted packets. As their names imply, LIJ jams the listen interval of a S-MAC schedule; CIJ jams the control interval; DPJ waits for a CTS packet and jams the ensuing data packet. We classify LIJ, CIJ and DPJ as detailed knowledge attacks, PCJ and RPCJ as minimal knowledge attacks. We suggest

Table 2: Average censorship rates and lifetime advantages of jamming attacks against S-MAC, LMAC and B-MAC (results are rounded, standard deviations in parenthesis).

D	N_j/N_s	S-MAC							LMAC				B-MAC		
		Rand.	React.	LIJ	CIJ	DPJ	PCJ	RPCJ	Rand.	React.	PSJ	RPSJ	Rand.	React.	LPLJ
Censorship rate R_c (%)															
15	0.75	92 (15)	93 (22)	48 (101)	93 (5)	79 (17)	75 (17)	59 (17)	78 (13)	94 (5)	96 (3)	80 (9)	85 (19)	93 (9)	93 (9)
	1.00	99 (1)	100 (0)	75 (36)	94 (5)	87 (8)	83 (8)	71 (8)	86 (10)	96 (5)	95 (4)	88 (8)	97 (9)	99 (2)	99 (2)
20	0.75	99 (2)	95 (14)	64 (63)	91 (6)	83 (13)	83 (9)	69 (11)	70 (18)	97 (4)	94 (5)	82 (15)	93 (11)	97 (5)	97 (5)
	1.00	100 (0)	100 (0)	89 (22)	94 (5)	91 (6)	83 (7)	76 (10)	91 (7)	97 (5)	97 (1)	91 (9)	97 (5)	99 (3)	99 (3)
Lifetime advantage R_l (%)															
15	0.75	35 (2)	17 (1)	124 (45)	188 (8)	35 (3)	50 (3)	43 (4)	14 (1)	17 (1)	32 (6)	41 (7)	24 (5)	22 (4)	161 (50)
	1.00	37 (2)	17 (1)	116 (28)	196 (11)	35 (2)	47 (3)	43 (2)	14 (1)	17 (1)	31 (4)	35 (3)	24 (5)	22 (3)	139 (31)
20	0.75	35 (1)	17 (1)	124 (31)	169 (11)	35 (4)	46 (3)	42 (3)	14 (1)	16 (1)	32 (5)	37 (4)	20 (3)	20 (3)	134 (37)
	1.00	37 (1)	17 (1)	113 (27)	173 (11)	34 (2)	44 (3)	43 (3)	14 (1)	16 (1)	29 (5)	34 (4)	20 (3)	19 (3)	126 (29)

for future work to compare the latest minimal knowledge attacks with existing detailed knowledge attacks [21].

Censorship rate Random jamming and reactive jamming have the highest censorship rates. Among the energy-efficient attacks, CIJ has the highest censorship rate. LIJ has large standard deviations because in LIJ, the SYNC packets are jammed – jammer motes that have too few sensor nodes or too many fellow jammer motes as neighbors would take a long time, if at all, to get hold of two SYNC packets to synchronize with the S-MAC schedule – making the censorship rate of LIJ highly topology-dependent. The censorship rate of DPJ is lower than CIJ's because of the adaptive listening mechanism in S MAC [21]. PCJ and RPCJ are not as effective as the detailed knowledge attacks (LIJ etc.), but their censorship rates are still substantial, at 83% and 76% respectively when $N_j/N_s = 1$ and $D = 20$. RPCJ is worse than PCJ because in the proactive approach of PCJ, the victim nodes go to sleep when they fail to access the jammed medium, whereas RPCJ misses more packets as a result of misalignment with the S-MAC schedule.

Lifetime advantage Table 2 agrees with the result of our previous work [21] that among the detailed knowledge attacks, CIJ has the highest lifetime advantage, followed by LIJ. The minimal knowledge attacks, PCJ and RPCJ, have, not surprisingly, lower lifetime advantages. DPJ, Random jamming and reactive jamming have the lowest. This means, in the absence of detailed knowledge, PCJ and RPCJ are genuine threats.

At high network density, the lifetime advantages of PCJ and RPCJ are comparable. The lifetime advantage of PCJ however decreases with network density, because as a jammer gets more neighbors, it fakes more transmissions, incurring a higher effort ratio. The lifetime advantage of RPCJ on the other hand hardly changes with network density, because an RPCJ jammer spends more time listening than transmitting, but the time spent on listening is roughly the duration of the listen interval, which does not change with network density.

A random jammer transmits for half of the time, and sleeps for half of the time. A reactive jammer transmits *at most* 10% of the time, and listens for at least 90% of the time, due to the 10% duty cycle. Based on these values alone, comparing the energy consumptions using Equation 10 tells us that reactive jamming has a lower effort ratio. However, random jamming has a far higher attrition rate, because it makes some victim nodes stay in the backoff state in the listening mode throughout the sleep interval, and therefore has a higher lifetime advantage than reactive jamming does.

To summarize, the high censorship rate and high lifetime advantage of CIJ indicates the importance of link-layer encryption. However, encryption alone is insufficient as the temporal arrangement of the packets induced by the nature of the protocol still allows PCJ and RPCJ to be effective and energy-efficient.

6.2 LMAC

We compare random jamming, reactive jamming with PSJ and RPSJ.

Censorship rate The censorship rates of PSJ are impressively close to those of random and reactive jamming. RPSJ is worse than PSJ, but both PSJ and RPSJ are more effective against LMAC than PCJ and RPCJ are against S-MAC, because the slot size can be estimated more accurately than the period of an S-MAC schedule.

Lifetime advantage Both PSJ and RPSJ have twice the lifetime advantages of random jamming and reactive jamming. Looking more carefully, RPSJ has a higher lifetime advantage than PSJ. This is because not all slots in a frame are necessarily

occupied, and when a slot is unoccupied, the energy RPSJ spends on listening is lower than the energy PSJ spends on transmitting.

As the network becomes denser, more slots in a frame are occupied, the effort ratios of both PSJ and RPSJ become higher, and hence their lifetime advantages decrease with network density. This trend should stop when all the slots in a frame are occupied.

Random jamming does not make LMAC nodes monitor the channel more than it usually does, unlike the case with S-MAC – the sensor nodes only listen for at most a small fraction of the slot size, and as packets are jammed, less energy is used on propagating the packets to the sink, resulting in a negative attrition rate. One note of caution though: had the jamming been allowed to start *before* the LMAC nodes synchronize with each other, the results would have been different, because then the LMAC nodes would have had to constantly listen for broadcast schedules, and this would have resulted in a large positive attrition rate, and hence lifetime advantage. Since reactive jamming transmits at most 2 bytes (one byte to jam a control packet, another byte to jam a data packet), or 174 μs per 20 ms slot, it does not satisfy Equation 10 and hence has a lower effort ratio than random jamming. In fact, reactive jamming will only have a higher effort ratio when the slot size is 1 ms according to Equation 10 which is unrealistic. With their attrition effect being similar, reactive jamming has a higher lifetime advantage than random jamming.

To summarize, PSJ and RPSJ have high censorship rates and lifetime advantages. Coupled with ease of implementation, they are genuine threats to LMAC even when the packets are encrypted.

6.3 B-MAC

We compare random jamming, reactive jamming with LPLJ. LPLJ is by design reactive jamming with optimized listening, so it is only intuitive that LPLJ has similar censorship rates as reactive jamming's, but far higher lifetime advantages than reactive jamming's. The lifetime advantage of LPLJ however decreases with network density due to the following reason.

As the network gets denser, a jammer mote gets not only more sensor nodes but also more jammer motes as its neighbors. More neighboring sensor nodes mean more packets to jam. More neighboring jammer motes means staying awake more often because whenever a signal is detected on the channel, a jammer mote always stays awake for a while to listen for a valid preamble. Consequently, a jammer mote has a higher effort ratio in a denser network, and according to Equation 11 and Equation 12, the lifetime advantage becomes lower. The lifetime advantages of random jamming are comparable to reactive jamming's, i.e. significantly lower than those of LPLJ.

To summarize, since LPLJ is trivial to implement and yet allows the jammer motes to live as long as, or longer than the victim nodes, it is a devastating threat to B-MAC even when the packets are encrypted.

7 Implications to Other Protocols

We now look at the implications of the above findings to other protocols. As explained in Section 3, we only concentrate on the MAC protocols that (1) use a single channel, and (2) listen instead of follow some schedule to receive messages. Among these protocols, there are protocols that use (1) slots, (2) frames or (3) random access to organize medium accesses, using the taxonomy in [20]. Examples, that use slots, frames

and random access, are S-MAC, LMAC and B-MAC respectively, which we have just investigated. We now look at other protocols that belong to each of the three categories, starting with slot-based protocols. We pick these protocols from [20].

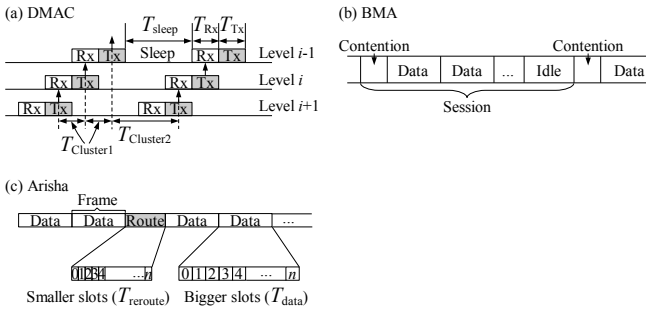


Figure 8: Medium access organization in (a) DMAC, (b) BMA, and (c) Arisha.

7.1 Slot-Based Protocols

Slot-based protocols include T-MAC [39] and DMAC [23]. Since T-MAC is derived from S-MAC, PCJ and RPCJ are applicable to T-MAC. The fact that T-MAC has a dynamic duty cycle offers some relief because even though PCJ and RPCJ are able to adapt to the dynamic duty cycle through periodic readjustments, they would be less effective and less efficient against T-MAC than against S-MAC due to the need for more frequent readjustments.

Like T-MAC, DMAC is a slotted protocol that uses a dynamic duty cycle, but unlike T-MAC, it offsets the schedule of a node depending on the number of hops the node is away from the sink, in order to minimize latency (Figure 8(a)). For example, if the node is i hops away, its schedule is offset by $+T_{\text{Rx}}$ compared with the schedule of a node $i + 1$ hops away. The following cluster pattern should emerge on the probability distribution of the packet interarrival times: Cluster1 has a mean of T_{Tx} and Cluster2 has a mean of T_{sleep}. Based on these clusters, PCJ and RPCJ can then be applied.

7.2 Frame-Based Protocols

Frame-based protocols include PACT [30], Arisha [4], TRAMA [32], BMA [22] and SS-TDMA [18]. These are TDMA protocols. The primary means of jamming these protocols is by way of estimating the slot size. To estimate the slot size, the jammer needs to be able to observe two consecutive slots, that is, the number of slots and the number of occupied slots in a frame need to satisfy Equation 2. This requirement is typically satisfied as explained in Section 4.2 and will not be mentioned again in the discussion below.

We start with SS-TDMA. SS-TDMA relies on the nodes being arranged in a rectangular or hexagonal rid. Due to the lack of control packets, SS-TDMA is easier to attack than LMAC since Equation 4 alone allows us to estimate the slot size.

Since both PACT and BMA use clustering to distribute TDMA schedules and both use similar frame structures, we only discuss BMA and extend our findings for BMA to PACT. In BMA, the network is partitioned into clusters. Every network starts with the cluster setup phase, where every node decides whether to become a clusterhead. At

the end of this phase, clusters are formed and the network enters the steady state phase. By this time, every clusterhead already knows the number of members in its cluster. The steady state phase is a sequence of sessions or frames. A session in turn consists of a contention period, a data transmission period and an idle period, and each of these periods are slotted (Figure 8(b)).

The so-called contention period allows the nodes to tell the clusterhead, in their assigned slot, their transmission schedules. Since the clusterhead already knows the number of nodes in the cluster, n, the number of slots in the contention period is exactly n. The data transmission period and idle period have a variable number of slots, but each session has a fixed length so that when there is more data to send, the number of data transmission slots is increased while the number of idle slots is decreased accordingly. Since the control packets and the data packets occupy a separate slot of their own, Equation 4 can be applied to estimate $T_{\text{contention}}$ and T_{data}. In the contention period, interarrival times are shorter than those in the data transmission period. This is how the jammer can tell whether it is in the contention period or the data transmission period. PSJ and RPSJ can now be applied to jam the contention period and the data transmission period, each with a different estimated slot size.

Arisha partitions the network into clusters, and in each cluster there is a gateway that arbitrates medium access among sensors and sets routes for sensor data. Arisha divides time into phases, the most important of which are the data transfer phase and reroute phase (Figure 8(c)). A phase is in turn made of frames. A data transfer frame has a bigger slot size than a reroute frame, i.e. $T_{\text{data}} > T_{\text{reroute}}$. A similar approach to jamming BMA, as explained before, can be applied to jamming Arisha.

TRAMA divides time into alternating periods of random access and scheduled access. Both periods are slotted. A slot in the random access period is called a signalling slot, while a slot in the scheduled access period is called a transmission slot. Denote the size of a signalling slot as $T_{\text{signalling}}$, and the size of a transmission slot as T_{Tx}, then for ease of synchronization, T_{Tx} is typically a multiple $T_{\text{signalling}}$, e.g. $T_{\text{Tx}} = 7T_{\text{signalling}}$ [32]. Again, a similar approach to jamming BMA can be applied to jamming TRAMA.

7.3 Random Access-Based Protocols

Random access-based protocols include low power listening [11], PCM [15], Sift [14] and WiseMAC [10].

Low power listening is to all our intents and purposes equivalent to B-MAC, so we will not discuss it any further. PCM is an improvement to IEEE 802.11 by exercising power control on a per-packet basis. Sift is a contention window-based MAC protocol. Since they do not specify any duty cycle, they offer no obvious exploits to PCJ, RPCJ, PSJ, RPSJ or LPLJ. But since they do not have any duty cycle, it remains to be seen how suitable they are, or how they can be adapted for WSNs.

WiseMAC uses the same preamble sampling scheme as B-MAC, with the difference being if the sender knows the schedule of the receiver, it waits until when the receiver is about to wake up and sends its packet with a normal, shorter preamble, instead of an LPL preamble. However, for broadcasting packets, the sender often has to stretch the preamble to the full length of the LPL preamble [20]. Therefore given enough broadcast traffic, the jammer is still able to figure out the check interval and apply LPLJ.

7.4 Discussion

Summing up, all protocols from Langendoen et al.'s survey [20] that we have discussed have weaknesses due to their organization of medium accesses. Among these protocols, frame-based protocols have better resistance to energy-efficient jamming because they spread out transmissions in time. Fortunately, more frame-based protocols than any other type of protocols have been proposed in literature. In the next section, we explore some countermeasures.

8 Countermeasures

In this section, we discuss potential countermeasures against attacks for each of the categories of MAC protocols.

8.1 S-MAC

Since our attacks against S-MAC are based on clustering, a countermeasure would naturally be to prevent clustering-based analysis from being feasible. This can be done by narrowing the distance between Cluster1 and Cluster2. Assuming no packet is transmitted in the sleep interval, the biggest possible Cluster1 interarrival time is the length of the listen interval, whereas the smallest possible Cluster2 interarrival time is the length of the sleep interval. We can 'stick' Cluster1 and Cluster2 together by equating the biggest Cluster1 interarrival time to the smallest Cluster2 interarrival time, which is tantamount to setting the duty cycle to 50%.

According to simulations with $N_j/N_s = 1$ and $D = 20$, this defence is modestly effective against PCJ if K-means is used as the clustering algorithm. The resultant censorship rate is reduced to 38% (with standard error 17%). However if expectation maximization (EM) [8] is used as the clustering algorithm, the censorship rate can only be reduced to 75% (with standard error 4%).

Relief can be found in the fact that a jammer using EM would considerably deteriorate its own lifetime advantage because EM is a computationally expensive and hence energy-consuming algorithm that involves multiple exponentiations. On the other hand, a duty cycle of 50% may not be energy-efficient enough for most WSNs that are characterized by low data rate [31]. Overall, using a high duty cycle is a partial countermeasure to energy-efficient link-layer jamming.

8.2 LMAC

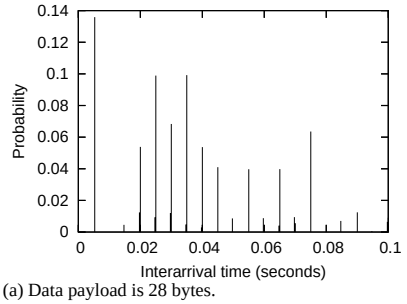
In the case of LMAC, we mentioned that it is advantageous to spread transmissions out in time, but as long as we transmit at fixed slot sizes (i.e. fixed intervals), spikes would manifest on the probability distribution graph of the packet interarrival times. The strategy of flattening the spikes to increase the difficulty in estimating the slot size can be served by changing the slot size pseudo randomly as a function of time and a hidden seed. Note that this method has a negative impact on the bandwidth of the protocol.

For example, if the sensor nodes change their slot size every second, by pseudo randomly picking a value from the range [20 ms, 30 ms], then according to simulations, a jammer using PSJ can still achieve a censorship rate of 80% (with standard error 12%) and a lifetime advantage of 47% (with standard error 5%) when $N_j/N_s = 1$ and $D = 20$.

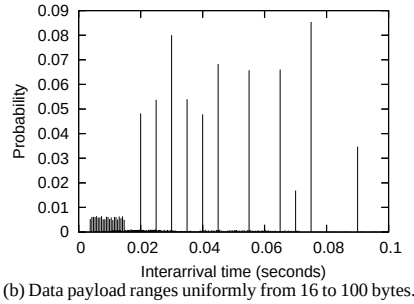
When we change the slot size – on a per slot basis– by incrementing the size with 25% until two times the original size is reached and then decrement with 25% until the original size again, the censorship rate of a PSJ jammer remains 80% (with standard error 14%), but the lifetime advantage decreases to 40% (with standard error 5%). The distribution of packet interarrival times is shown in Figure 9.

Remarkable of this countermeasure is the RPSJ censorship: only 35% (with standard error 15%). The reason for this effect is that the RPSJ method of attacking listens for a preamble before jamming, a jammer is only efficient when it starts to receive *just before* a packet is transmitted. When it wakes during the transmission of a packet or after – what we try to achieve with this countermeasure– it waits in receive mode until the next preamble, hence wasting energy. The PSJ attack does not suffer from this effect.

Note that the above described countermeasure has deterministic behavior, which makes it less suitable. However, in Section 9 we implement it on sensor node prototypes to show its effect in practice.



(a) Data payload is 28 bytes.



(b) Data payload ranges uniformly from 16 to 100 bytes.

Figure 9: Probability distribution of packet interarrival times for LMAC with dynamic change of slot size

The above described countermeasures are less than satisfactory. We succeed in fouling the attackers in the estimations they make of the slot size by flattening out spikes in the probability distribution of interarrival times. In the total picture, however, this misses its effect. The fact that PSJ attackers have more than one opportunity to jam a packet – due to the multi-hop communication to the sink node– makes them robust and effective in jamming.

In the following scenario, we make the LMAC protocol more persistent in delivering messages by doubling the number of retries the LMAC protocol does before discarding packets. As a matter of fact, LMAC without countermeasure and the number of retries set to 6, has no effect on censorship rate. With countermeasure, however, the PSJ censorship rate R_c drops to 65% (with standard error 14%). When we further decrease the effect of the attackers by reducing them in number, the censorship rates drop to 51% (with standard error 9%) and $R_c = 38\%$ (with standard error 25%), for $N_j/N_s = 0.75$ and $N_j/N_s = 0.50$ respectively. The conclusion is that an attacker should not be thrifty with deploying jammer nodes.

In the above discussed results, the data packet payload ranges uniformly from 16 to 100 bytes. Large payloads give the PSJ attacker more opportunity (i.e. higher probability) to jam the packet. We simulate attacks with fixed data packet payload of 28 bytes. The censorship rates for PSJ are $R_c = 49\%$ and $R_c = 29\%$, for $N_j/N_s = 1.00$ and $N_j/N_s = 0.75$ respectively. Thus to avoid effective attacks, shorter data packets

should be preferred.

8.3 B-MAC

For B-MAC, it is not clear how to prevent attacks, because B-MAC relies on the preamble to be long enough for the receivers to detect. Shortening the preamble any further than what we have simulated, 10 ms, which is the minimum considered by Polastre et al. [31], defeats the purpose of B-MAC.

8.4 Discussion

From the above discussion, it appears for now that an effective countermeasure is lacking, but by comparing the censorship rate and lifetime advantage of the best attack on the respective protocols, LMAC emerges as a better choice than S-MAC and B-MAC in terms of resistance against link-layer jamming. Generalizing this observation, TDMA protocols are potentially a better choice than other types of protocols.

This concludes the presentation of our simulated attacks and countermeasures for the three main classes of MAC protocols.

9 Validation

To validate our simulation results, we implement LMAC and its most effective jammer (PSJ) on prototype sensor node hardware.

9.1 Implementation

For practical reasons, the following changes with respect to our simulation model are considered. Firstly we test the effectiveness of jamming and countermeasure in a single hop network. The jammer node is in radio range of the sink node and therefore we expect the censorship rate to be 100% for an effective jammer.

Secondly, we consider a three-node network (i.e. one sink node and two well-behaved nodes), which uses a frame length of four slots (i.e. the probability of two consecutive slots is 1). Each slot takes $1/16$s, roughly a factor 3 larger than in simulation[2], but is more practical to implement, since it allows for sufficient time to process the packets and to prepare the transceiver to transmit the data. These steps are not considered in our simulations, but cannot be omitted on our prototype sensor nodes.

Thirdly, Our prototypes are based upon the hardware described in [41], however the RFM TR1001 transceiver is exchanged for a Nordic nRF905 transceiver [28] for the following reason. The Nordic transceiver releases the processor from time critical controlling, like the generation of the preamble and the alignment of the incoming bit stream. This results in a better performance of the communication between nodes. However, the transceiver has slightly different properties as are assumed in our simulation model. The transceiver denies us the opportunity to transmit the short jamming packets as proposed in Section 5. The jamming is done by transmitting a packet consisting of a preamble, address, payload of 1 byte and a checksum. A smaller packet cannot be transmitted by the transceiver hardware.

Finally, we let the PSJ observe the packet interarrival times for control packets only, due to two reasons: (1) the used transceiver filters the packets on type, length and

[2]For the simulations in this section, we use timing as discussed here.

validity before handing over to the processor and (2) our countermeasure for LMAC leaves the gap T_{gap} between control and data packet untouched. Therefore an attacker might estimate this gap accurately, since it would show up in the probability distribution of packet interarrival times and might –based on this information– distinguish between the two packet types. We let the PSJ observe 8 interarrival times before letting it estimates the slot size.

The implementation of the PSJ attacker consumes –on top of the DCOS kernel and drivers [12]– 1520 bytes of program memory including debugging code that allows us to get insight in the estimated slot interval via serial link to the jamming node.

In our experiments, the sink node keeps track of correctly received data packets by recording their sequence number. This allows us to detect missing packets and thus to calculate censorship rate R_c. Both remaining well-behaved nodes transmit 16 byte data packets every second. When a data packet is not acknowledged by the sink node, it is scheduled for retry. However, when it fails to be delivered within three retries, the packet is simply discarded. In a test without attacker, we verified that all generated data packets arrive in the sink node.

9.2 Experiments

First, we test the effect of jamming in on a LMAC setting without countermeasures. The experiments show that jamming is effective; no communication between nodes and sink is possible when the jammer is active (i.e. $R_c = 100\%$). Plugging in a similar setting in our simulation model, results in $R_c = 97\%$. A closer inspection shows that the 3% difference is caused by the packets that reach the sink before the simulated PSJ attacker has established an estimate of the slot size.

When the PSJ estimates the slot size incorrectly, the nodes are able to communicate again after the transmission of the jammer has drifted to the gap between data packet and the control packet of the next slot, missing its jamming effect. This effect occurred in one of ten experiments. It implies that the jammer node must resynchronize frequently, to ensure effective jamming. Our simulation model does not account for drifting.

Another observation that we make is the following. When the nodes are closely situated together and the attacker is relatively far away from the cluster, the LMAC packets are *not* disrupted by the jammer node while it is still in radio range. Apparently, the demodulation scheme of the transceiver is robust against interfering signals with lower signal strength. This makes it harder for jammers to compromise a wireless sensor network, since it requires them to transmit at high power level or to use specialized hardware. As we already remarked in Section 5, we omitted interference and the gray area effect in our simulations, however they have a clear influence on the results.

In the next experiment, LMAC is extended with the countermeasure discussed in Section 8.2. For this purpose, we extend the control packet of LMAC with one byte used as 'hidden' seed for the countermeasure. At the end of each slot, this value is incremented by one, advancing to the next table entry that contains the 25% increment and decrement steps of the slot size. By receiving one control packet correctly, a node is able to calculate all future steps of the countermeasure scheme.

Compared to the simulations in Section 8.2, our implementation on sensor nodes uses only a small portion of the slot to actually transmit the control and data packets. This severely reduces the probability that an attacker jams a packet, certainly when its estimate of the slot size does not match reality. Therefore, our countermeasure against the PSJ attack is –in this setting– quite effective. We observe a censorship rate of

$R_c = 2\%$ (32 of 1532 data packets were successfully jammed). In the trace of successful received messages in the sink, we see that the jammer is sometimes successful in disturbing a packet, however, in most cases the acknowledgement mechanism in LMAC is able to recover the packet in one of the retries. Similar effects are observed in the simulation results of this scenario. Our simulator model reported a censorship rate of $R_c = 0\%$.

9.3 Discussion

Our experiments validate our simulations on the following aspects: (1) packets can be nullified with short jamming packets and (2) censorship rates match with the experiments in a small network. Although, the validation of the simulation model is not complete, some interesting points have been brought to our attention: (1) processing delays, (2) radio aspects (i.e. demodulation scheme, gray area and interference) and (3) timing imperfections of jammers cannot be neglected. These effects are not taken into account in our simulation model. It is our future work to extend and validate the simulation model in more detail.

In addition, we showed that LMAC without countermeasure can be effectively be jammed. This shows the importance of keeping potential link-layer attacks in mind when designing a MAC protocol.

10 Related Work

Wood et al. wrote in 2002 that no effective defence was yet known against link-layer jamming [46]. Both Ståhlberg [37] and Wood et al. [46] quote how an attacker might keep sending RTS packets to elicit CTS packets from its victims.

Negi and Perrig [27] investigate the attack of a jammer that detects and jams RTS packets, as well as sends RTS packets to reserve the largest time interval possible, using the Poisson arrival model. This type of jammer needs to be an insider of the network, in order to know the content of the packets, and also send RTS packets that can pass the integrity check by the normal sensor nodes. Our attackers are not bound by such an assumption. Moreover, if the RTS packet is short enough, the jammer may not have enough time to respond [47]. Jamming the *ensuing* CTS or data packets might be a more successful attack [21].

Konorski [17] proposes a scheduling policy that addresses the selfish behavior of using small or no contention time (also called random backoff time) in contention-based protocols, in the context of *single-hop* networks. We are only interested in multi-hop networks. Kyasanur et al. [19] propose that instead of letting the sender set the contention time, the receiver sets and sends the time in the CTS and ACK packets to the sender. The sender uses this assigned contention time in the subsequent transmission to the receiver. The receiver can then tell if the sender is being selfish, i.e. using a value smaller than the assigned value, by observing the number of idle slots between consecutive transmissions from the sender. The problem with this approach is that if the receiver misbehaves, the sender is penalized. Cárdenas et al. [6] propose using Blum's coin flipping protocol [5] to ensure that the sender and the receiver choose a random backoff time, such that if *either* the sender or the receiver deviates from the random backoff, the other party would know. Čagalj et al. [44] investigate the effect of a group of selfish cheaters from a game-theoretic viewpoint. We focus on DoS attacks,

in which the purpose of the misbehaving party lies in disruption instead of getting more bandwidth. Different countermeasures are thus required.

Xu et al. [48] propose two evasion strategies against constant jammers: (1) channel surfing and (2) spatial retreat. Channel surfing is essentially an adaptive form of frequency hopping. Instead of continuously hopping from frequency to frequency, a node only switches to a different frequency when it discovers the current frequency is being jammed. Spatial retreat is an algorithm according to which two nodes move in Manhattan distances to escape from a jammed region. The algorithm is more suitable for wireless sensor and actor networks (WSANs) [2] than conventional WSNs. The algorithms are effective against constant jammers but we are motivated to look at jammers who are more intelligent than constant jammers.

We mentioned the 4 generic jammer models by Xu et al. [47] earlier. Based on the models, Xu et al. show that either received signal strength indication (RSSI) or carrier sensing time alone is not sufficient in detecting all the 4 types of jammers. Instead, attacks can be detected by measuring (1) both the packet delivery ratio and the signal strength, or (2) both the packet delivery ratio and the location. In our previous work [21], we look at potential attacks on one particular MAC protocol, which is S-MAC, and provide a countermeasure.

Instead of looking at the packet delivery ratio of individual nodes, we look at the effect of distributed jammer motes on the WSN as a whole. In our opinion, apart from jamming effectiveness, a jammer cares about jamming efficiency. For this reason, while Xu et al. provide metrics for measuring the quality of service of individual nodes, we provide metrics for measuring the jamming effectiveness and efficiency of the jammers. Xu et al.'s and our work are complementary in the sense that Xu et al. target jammers at the physical layer, while we target jammers that aim to gain more edge by exploiting the data link layer. This work extends our previous work [21] to more protocols and in a more general setting (the latest attacks work even on encrypted traffic, while the earlier attacks do not).

11 Conclusion

In earlier work we investigated link-layer jamming based on *detailed* knowledge of the MAC protocols. In the present paper we propose new link-layer jamming algorithms that are based on *minimal* knowledge of the target protocols. The effectiveness and energy-efficiency of the minimal knowledge attacks is almost as good as the effectiveness and energy-efficiency of the detailed knowledge attacks. In addition, the minimal knowledge attacks are effective when data packets are encrypted. We propose new metrics (censorship rate, attrition rate, effort ratio and lifetime advantage) and a method to quantify the effect of link-layer jamming.

We implemented LMAC and its most effective jammer on sensor node prototype hardware to validate our simulation model. The censorship rates matched our simulations in a four-node topology. With our experiments we pinpointed that (1) timing aspects, like drift in jammer motes, and (2) RF aspects (i.e. demodulation scheme, gray area and interference) deserve more attention in our model.

With typical WSN systems in use today no effective measures against link-layer jamming are possible. For WSNs that require high security against link-layer jamming we recommend (1) encrypting link-layer packets to ensure a high entry barrier for jammers, (2) the use of spread spectrum hardware, (3) the use of a TDMA protocol, and (4) the use of randomized transmission intervals. It is our future work to establish

analytically the desirable characteristics of MAC protocols that are secure against link-layer jamming attacks.

Acknowledgements

The authors would like to thank Ferdy Hanssen and Tjerk Hofmeijer for their help with experiments.

References

[1] D. L. Adamy and D. Adamy. *EW 102: A Second Course in Electronic Warfare*. Artech House Publishers, 2004.

[2] I. Akyildiz and I. Kasimoglu. Wireless sensor and actor networks: Research challenges. *Elsevier Ad Hoc Networks*, 2(4):351–367, Oct. 2004.

[3] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.

[4] K. Arisha, M. Youssef, and M. Younis. Energy-aware TDMA based MAC for sensor networks. In *IEEE Workshop on Integrated Management of Power Aware Communications Computing and Networking (IMPACCT 2002)*, May 2002.

[5] M. Blum. Coin flipping by telephone a protocol for solving impossible problems. *SIGACT News*, 15(1):23–27, 1983.

[6] A. A. Cárdenas, S. Radosavac, and J. S. Baras. Detection and prevention of MAC layer misbehavior in ad hoc networks. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 17–22, New York, NY, USA, 2004. ACM Press.

[7] S. Chatterjea, L. van Hoesel, and P. Havinga. AI-LMAC: An Adaptive, Information-centric and Lightweight MAC Protocol for Wireless Sensor Networks. In *Proceedings of the Intelligent Sensors, Sensor Networks and Information Processing Conference*, pages 381–388, IEEE Computer Society Press, 2004.

[8] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39(1):1–38, 1977.

[9] A. El-Hoiydi. Aloha with preamble sampling for sporadic traffic in ad hoc wireless sensor networks. In *IEEE International Conference on Communications*. IEEE Press, pages 3418–3423, Apr. 2002.

[10] A. El-Hoiydi, J.-D. Decotignie, C. Enz, and E. L. Roux. Poster abstract: WiseMAC, an ultra low power MAC protocol for the WiseNET wireless sensor network. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 302–303, New York, NY, USA, 2003. ACM Press.

[11] J. Hill and D. Culler. Mica: a wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, Nov. 2002.

[12] T.J. Hofmeijer, S.O. Dulman, P.G. Jansen, and P.J.M. Havinga. DCOS, A Real Time, Leight-Weight Data Centric Operating System. In *IASTED Int. Conf. on Advances in Computer Science and Technology (ACST)*, pages 259–264, 2004.

[13] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11(1):2–16, 2003.

[14] K. Jamieson, H. Balakrishnan, and Y. Tay. Sift: A MAC protocol for event-driven wireless sensor networks. Technical Report MIT-LCS-TR-894, Massachusetts Institute of Technology, May 2003.

[15] E.-S. Jung and N. H. Vaidya. A power control MAC protocol for ad hoc networks. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 36–47, New York, NY, USA, 2002. ACM Press.

[16] C. Karlof, N. Sastry, and D. Wagner. TinySec: a link layer security architecture for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 162–175, New York, NY, USA, 2004. ACM Press.

[17] J. Konorski. Multiple Access in Ad-Hoc Wireless LANs with Noncooperative Stations. In *NETWORKING 2002: Proceedings of the Second International IFIP-TC6 Networking Conference on Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; and Mobile and Wireless Communications*, volume 2345 of *LNCS*, pages 1141–1146. Springer-Verlag, 2002.

[18] S. S. Kulkarni and U. Arumugam. TDMA Service for Sensor Networks. *Proceedings of the Third International Workshop on Assurance in Distributed Systems and Networks (ADSN)*, pages 604–609, Mar. 2004.

[19] P. Kyasanur and N. Vaidya. Detection and Handling of MAC Layer Misbehavior in Wireless Networks. In *Int. Conf. on Dependable Systems and Networks (DSN'03)*, pages 173–182. IEEE Computer Society Press, 2003.

[20] K. Langendoen and G. Halkes. In *Embedded Systems Handbook*, Chapter 34: Energy-Efficient Medium Access Control. Editor: R. Zurawski, CRC Press, 2005, ISBN 0-8493-2824-1.

[21] Y. Law, P. Hartel, J. den Hartog, and P. Havinga. Link-layer jamming attacks on S-MAC. In *2nd European Workshop on Wireless Sensor Networks (EWSN 2005)*, pages 217–225. IEEE, 2005.

[22] J. Li and G. Y. Lazarou. A bit-map-assisted energy-efficient MAC scheme for wireless sensor networks. In *IPSN'04: Proceedings of the third international symposium on Information processing in sensor networks*, pages 55–60, New York, NY, USA, 2004. ACM Press.

[23] G. Lu, B. Krishnamachari, and C. S. Raghavendra. An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks. *18th International Parallel and Distributed Processing Symposium (IPDPS'04) - Workshop 12*, 13(13), 2004.

[24] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97. ACM Press, 2002.

[25] D. McKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[26] M. Mysore, M. Golan, E. Osterweil, D. Estrin, and M. Rahimi. TinyDiffusion in the Extensible Sensing System at the James Reserve. `http://www.cens.ucla.edu/~mmysore/Design/OPP`, University of California, Los Angeles, U.S.A., May 2003.

[27] R. Negi and A. Perrig. Jamming analysis of MAC protocols. Carnegie Mellon Technical Memo, 2003.

[28] Nordic Semiconductor ASA. Single Chip 433/868/915 MHz Transceiver nRF905. Datasheet, V1.2, 2005.

[29] G. Noubir. On connectivity in ad hoc networks under jamming using directional antennas and mobility. In *Wired/Wireless Internet Communications*, volume 2957 of *LNCS*, pages 186–200. Springer-Verlag, 2004.

[30] G. Pei and C. Chien. Low power TDMA in large wireless sensor networks. In *Military Communications Conference (MILCOM 2001)*, volume 1, pages 347–351. IEEE, 2001.

[31] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107. ACM Press, 2004.

[32] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves. Energy-efficient collision-free medium access control for wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 181–192. ACM Press, 2003.

[33] N. Reijers, G. Halkes, and K. Langendoen. Link layer measurements in sensor networks. In *1st IEEE Int. Conf. on Mobile Ad hoc and Sensor Systems (MASS '04)*, pages 224–234, IEEE Computer Society Press, 2004.

[34] RF Monolithics, Inc. *ASH Transceiver Software Designers Guide*, 2002.

[35] RF Monolithics, Inc. TR1001: 868.35 MHz Hybrid Transceiver. Datasheet, 2002.

[36] RF Monolithics, Inc. *ASH Transceiver Designers Guide*, 2004.

[37] M. Ståhlberg. Radio jamming attacks against two popular mobile networks. In H. Lipmaa and H. Pehu-Lehtonen, editors, *Proceedings of the Helsinki University of Technology. Seminar on Network Security. Mobile Security*. Helsinki University of Technology, Finnland, 2000.

[38] Texas Instruments, Inc. MSP430x13x, MSP430x14x Mixed Signal Microcontroller. Datasheet, 2001.

[39] T. van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 171–180. ACM Press, 2003.

[40] L. van Hoesel and P. Havinga. A Lightweight Medium Access Protocol (LMAC) for Wireless Sensor Networks: Reducing Preamble Transmissions and Transceiver State Switches. In *In 1st International Workshop on Networked Sensing Systems (INSS 2004)*,pages 205–208, June 2004.

[41] L.F.W. Hoesel, S.O. Dulman, P.J.M. Havinga, H.J. Kip. Design of a low-power testbed for Wireless Sensor Networks and verification. CTIT Technical report, September 2003.

[42] J. van Lint and R. Wilson. *A course in combinatorics*. Cambridge University Press, 2nd edition, 2001.

[43] K. Vantran. WolfPack Proves Strength in Numbers. DefenseLINK News, Aug. 2003.

[44] M. Čagalj, S. Ganeriwal, I. Aad, and J.-P. Hubaux. On Cheating in CSMA/CA Ad Hoc Networks. Technical Report IC/2004/27, École Polytechnique Fédérale de Lausanne, Suisse, 2004.

[45] A. S. Wander, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz. Energy analysis of public-key cryptography for wireless sensor networks. In *Third IEEE International Conference on Pervasive Computing and Communications (PERCOM'05)*, pages 324–328. IEEE Computer Society Press, 2005.

[46] A. Wood and J. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62, Oct. 2002.

[47] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of 6th ACM Inter. Symp. on Mobile Ad Hoc Networking and Computing (ACM MOBIHOC'05)*, pages 46–57. ACM Press, 2005.

[48] W. Xu, T. Wood, W. Trappe, and Y. Zhang. Channel surfing and spatial retreats: defenses against wireless denial of service. In *WiSe '04: Proceedings of the 2004 ACM workshop on Wireless security*, pages 80–89, New York, NY, USA, 2004. ACM Press.

[49] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC protocol for Wireless Sensor Networks. In *Proc. IEEE Infocom*, pages 1567–1576, New York, NY, USA, June 2002. USC/Information Sciences Institute, IEEE.

[50] W. Ye, J. Heidemann, and D. Estrin. Medium Access Control with Coordinated, Adaptive Sleeping for Wireless Sensor Networks. *IEEE/ACM Transactions on Networking*, 12(3):493–506, 2003.

[51] H. Zhang and J. Hou. On deriving the upper bound of α-lifetime for large sensor networks. In *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pages 121–132. ACM Press, 2004.

[52] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 1–13, New York, NY, USA, 2003. ACM Press.

Appendix

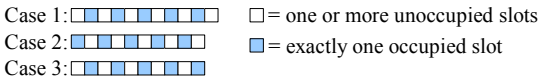


Figure 10: Three cases of slot distribution with no consecutive occupied slots.

Proof of Equation 1: Given s ($s \geq 4$) total number of slots and n ($0 \leq n \leq s$) occupied slots in a frame, we want to find the probability that at least two occupied slots are consecutive (denoted as event A), which is equivalent to 1 minus the probability that there is at least one unoccupied slot between any two occupied slots (denoted as event B). When $0 \leq n < 2$, $\Pr\{A\} = 0$, since there is either no occupied slot, or there is always one unoccupied slot between two occupied slots. When $\frac{s}{2} < n \leq s$, $\Pr\{A\} = 1$, since more than half of the slots are occupied and at least two of them are consecutive. For the other values of n, we refer to Figure 10 where one white box represents one or more unoccupied slots, and one shaded box represents exactly one occupied slot. Denote x_i ($x_i \geq 1$) as the number of unoccupied slots in the i-th white box. There are 3 cases to consider:

Case 1: $x_1 + \ldots + x_{n+1} = s - n$, $s \geq 2n + 1$, and the number of positive integer solutions for $x_1, \ldots, x_{n+1}$ is $\binom{s-n-1}{n}$ [42].

Case 2: $x_1 + \ldots + x_n = s - n$, $s \geq 2n$, and the number of solutions for $x_1, \ldots, x_n$ is $\binom{s-n-1}{n-1}$.

Case 3: Similar to Case 2, the number of solutions for x_1, …, x_n is $\binom{s-n-1}{n-1}$.

Therefore, when $2 \leq n \leq \frac{s-1}{2}$,

$$\Pr\{A\} = 1 - \Pr\{B\} = 1 - \frac{\binom{s-n-1}{n} + 2\binom{s-n-1}{n-1}}{\binom{s}{n}} = 1 - \frac{\binom{s-n}{n}}{\binom{s-1}{n}}$$

And when $\frac{s-1}{2} < n \leq \frac{s}{2}$, i.e. when $n = \frac{s}{2}$ and s is even,

$$\Pr\{A\} = 1 - \Pr\{B\} = 1 - \frac{2\binom{s-n-1}{n-1}}{\binom{s}{n}}$$

□