Link-layer Jamming Attacks on S-MAC

Yee Wei Law Pieter Hartel Jerry den Hartog Paul Havinga
Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente, Enschede, The Netherlands
Email: {ywlaw, pieter, j.i.denhartog, havinga}@cs.utwente.nl

Abstract

We argue that among denial-of-service (DoS) attacks, link-layer jamming is a more attractive option to attackers than radio jamming is. By exploiting the semantics of the link-layer protocol (aka MAC protocol), an attacker can achieve better efficiency than blindly jamming the radio signals alone. In this paper, we investigate some jamming attacks on S-MAC, the level of effectiveness and efficiency the attacks can potentially achieve, and a countermeasure that can be implemented against one of these attacks.

1. Introduction

Radio jamming is potentially the most direct, non-destructive and yet disruptive form of DoS attack on sensor networks. There are two reasons why an attacker might favor radio jamming over other DoS attacks: (1) it is trivial to execute – the jammer only needs to emit an arbitrary constant signal at a power roughly equal to the signal power of its victims [14], (2) sensor nodes are typically limited to single-frequency use [16], depriving them of the possibility to use standard countermeasures such as direct sequence spread spectrum and frequency hopping spread spectrum, or some combination of these two techniques [10]. On the other hand, blind radio jamming is not nearly as energy-efficient as link-layer jamming. Let us elaborate in the paragraph below.

Suppose there is a sensor network that has no fixed base stations as points of control (e.g. in a high-security application). A radio jammer, having *no* base stations to focus its jamming signals on, would have to target the sensor nodes themselves. Assume that every sensor node has a signal power of P_s. To jam an area of radius r using an omni-directional antenna, each jamming device needs to draw a power that is proportional to $r^\beta P_s$ ($3.5 \leq \beta \leq 5$ for outdoor environments) [10]. That is, the wider the area, the exponentially more power a jamming device needs. If the energy to these devices cannot be replenished, with high probability, the devices would exhaust themselves much faster than the sensor nodes would. A solution for the attacker is to reduce the jamming power and increase the number of jamming devices, to such an extent that the attacker would reduce to using a network of jamming motes, that is, a network of nodes having comparable capability with the sensor nodes. Taking this a step further, instead of blind radio jamming, the attacker can save energy with link-layer jamming, by taking advantage of the MAC protocol. Note that we are proposing a general attack strategy where

1. there are no fixed *sinks* (nodes that request for, and hence sink information) to attack, e.g. in directed diffusion [3] where ID-less interests propagate through the network, there is no way for a node to tell where the real sinks are;

2. it is infeasible for the attacker to deploy its nodes strategically, except perhaps to air-drop them on top of the target network;

3. the attacker can only estimate where and not how the target network is or would be deployed – knowing the location allows the attacker to plan an ambush though.

We will show in this paper how an attacker may take advantage of the MAC protocol, using S-MAC [17], one of the earliest protocols specifically designed for sensor networks, as our example. The contributions of this paper are (1) to show 3 easy-to-implement attacks on S-MAC, and (2) to show an equally easy-to-implement countermeasure against one of these attacks, thereby (3) delivering the message that MAC protocols should be designed with prevention against DoS attacks in mind.

The paper is organized as follows. Section 2 summarizes the S-MAC protocol. The attacks are described in Section 3, followed by a countermeasure in Section 4. Related work is discussed in Section 5 before the conclusion is given in Section 6.

2. The S-MAC protocol

The prime focus in the design of MAC protocols for sensor networks is minimizing energy use. S-MAC reduces energy wastage by avoiding collisions, overhearing, control packet overhead and idle listening. In S-MAC, every node follows one or more *schedules*. The node that first sets and broadcasts a schedule is the *synchronizer*. Other nodes that receive and adopt the schedule are *followers*. The synchronizer and followers thus form a *virtual cluster*, listening and sleeping at about the same time.

A schedule indicates the times t_0, $t_0 + T_P$, $t_0 + 2T_P$, ... at which the synchronizer and followers go to sleep, where T_P is the length of a cycle, or *period* (Figure 1). A complete cycle consists of a listen interval T_L seconds long and a sleep interval T_S seconds long, i.e. $T_P = T_L + T_S$. The *duty cycle* is defined as the ratio T_L/T_P, therefore the lower the duty cycle, the more energy nodes get to conserve, albeit at the price of incurring higher network latency. A listen interval is further divided into a synchronization (SYNC) interval, and a control (CTRL) interval.

The SYNC interval is for sending SYNC packets that allow neighboring nodes to synchronize their schedules:

1. When a node first joins a network, it listens for a whole period. If the channel is clear, the node broadcasts its schedule in a SYNC packet in the SYNC interval. A SYNC packet consists primarily of the address of the sender and the time the sender goes to sleep, t_s, relative to the sender's current time. This node becomes a synchronizer.

2. If a node receives a schedule from a neighbor before choosing its own schedule, it adopts the schedule, and re-broadcasts the schedule after a random delay t_d, indicating that it will sleep at $t_s - t_d$. This node becomes a follower.

3. If a node receives a different schedule after broadcasting its own, it adopts both schedules [18].

Every node broadcasts its schedule periodically (every *SYNC period*) even if it has not received any other node's schedule, so that when new nodes join the network, they are able to synchronize with existing schedules.

The CTRL interval is for medium reservation, to avoid collision during node-to-node data transmissions, by means of exchanging RTS and CTS packets, akin to the IEEE 802.11 Distributed Coordination Function (DCF). The CTRL interval is also for data broadcast, in which case no exchange of RTS/CTS packets is involved.

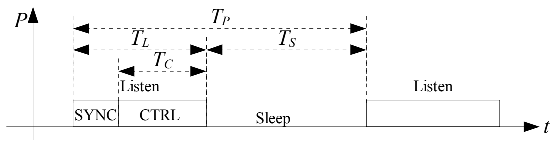


Figure 1. S-MAC schedule.

3. Attacks on S-MAC

In this section we discuss our assumptions of the sensor network and the attacker (as a network of jamming motes), details of our attacks and countermeasure, coupled with simulation results.

3.1. Assumptions

We assume an attacker has 2 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. While a sink node is a node that requests information from other sensor nodes, the nodes that supply the requested information are called source nodes [3].

We assume the sensor nodes exchange messages that are not encrypted although they might or might not be cryptographically signed. We make this assumption because: (1) encrypted messages introduce an additional cost of at least 200 cycles or 400 nJ per byte using the fastest symmetric algorithm on an ultra-low-power processor [6]; (2) we know of no key management scheme that protects the link layer with more than one encryption key, and consequently the compromise of a single key compromises the whole network; (3) by listening on the channel for some time an attacker can still learn about the S-MAC parameters and attack accordingly. The third point deserves elaboration: if we assume data packets to be typically larger than SYNC/RTS/CTS/ACK packets (which are roughly of the same length), an attacker might be able to build a prediction model of when data packets are transmitted, by just collecting statistics of the times when longer packets are transmitted. In fact, we are working on such a prediction model.

However we do not assume the attackers know the values of the S-MAC system parameters (e.g. the period, the duty cycle etc.) used by the sensor nodes. Our attacks include a method of estimating these parameters just by listening.

3.2. Simulation and evaluation model

All protocols, attacks and countermeasures have been simulated on Pentium 4 PCs with at least 768 MB of RAM using the OMNeT++ simulator (`www.omnetpp.org`). In the case of no attackers, 100 normal nodes (1 sink node + 99 source nodes) are pseudorandomly distributed in a square area whose length l is derived from the intended network density D [20] and transmission range r using Equation 1:

$$l = \sqrt{\frac{100\pi}{D}}r \tag{1}$$

The normal nodes use S-MAC with *adaptive listening* [18] and a duty cycle of 10% at the link layer, and TinyDiffusion [8] at the network layer, both faithfully ported from TinyOS (`tinyos.sf.net`). For a total of T_{sim} virtual seconds, the sink node broadcasts an interest every 10 seconds, while the source nodes each broadcast a matching data every 2 seconds, as an approximation of a network with moderately fast-changing data. T_{sim} is set to 600 and 1200, where the longer time is for seeing the effect of longer simulation time on the results. For simulating mobility, the improved random waypoint model [19] is used, with a minimum speed of $r/20$ and a maximum speed of $r/5$ (giving the average speed as $r/8$). We believe that the choices above reflect the typical workload of a small-scale sensor network [1]. Every experiment is run 20 times, using 32 new seeds for the 32 pseudorandom number generators in OMNeT++ each time. Attacks are simulated by adding 50 or 100 jamming motes, of the same transmission range, energy supply and power consumption as the normal nodes, to the network. The ratio between the power consumption in sleep, Tx and Rx mode is 1:2400:960 [12]. The source code is available at `wwwes.cs.utwente.nl/eyes/smac_attack.zip`.

One of the things that are not simulated is the interference resulted from jamming, which would in practice cause more data packet loss than in simulation. However this is one 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 [11].

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 percentage 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

$$R_c = (M - M')/M \times 100\% \tag{2}$$

R_a measures the percentage 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

$$R_a = (E' - E)/E \times 100\% \tag{3}$$

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. Using R_a and R_e, we can calculate the *lifetime advantage* R_l of a jamming mote over a sensor node, that is how long a jamming mote can live compared with a sensor node. Let T_{att}, T' be the lifetime of a jamming mote and the lifetime of a sensor node under attack respectively. Assuming the power is constant during the lifetime of a jamming mote or sensor node, i.e.

$$E_{\text{total}}/T_{\text{att}} = E_{\text{att}}/T_{\text{sim}} \qquad E_{\text{total}}/T' = E'/T_{\text{sim}}$$

then the attacker's lifetime advantage

$$R_l = T_{\text{att}}/T' = E'/E_{\text{att}} = (1 + R_a)/R_e \tag{4}$$

3.3. Details of attacks

Since S-MAC uses a periodic listen-sleep schedule, the simplest attack is to jam the entire listen interval. If the duty cycle is 50%, such an attack ideally allows the attacker to use 50% less energy compared with using radio jamming. Ironically, the more energy the network wants to conserve with a lower duty cycle, the more energy-efficient the attacker becomes. A jamming mote can use Equation 5 to 7 to estimate the period T_P, the length of a listen interval T_L and the length of a CTRL interval T_C, by just listening to SYNC packets (Figure 2):

$$T_P = t_1 + t_{s1} - t_0 - t_{s0} \tag{5}$$
$$T_L = \max(t_{s0}, t_{s1}, t_{s2}, ...) + \delta \tag{6}$$
$$T_C = \min(t_{s0}, t_{s1}, t_{s2}, ...) - \delta \tag{7}$$

where t_{s0}, t_{s1}, t_{s2} are sleep times obtained from SYNC packets, δ is a small estimated constant that compensates for the DCF Inter-Frame Space (DIFS) [2] and carrier-sensing time spent by the sender.

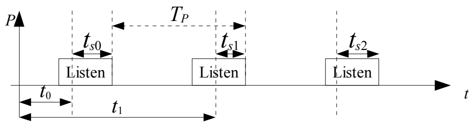


Figure 2. Estimating S-MAC parameters.

Once a jamming mote gets an estimation of T_P and T_L, it can adopt a periodic jam-sleep schedule, jamming when its neighbors are listening, and sleeping when its neighbors are sleeping. Note that since a jamming mote must watch out for a pair of consecutive SYNC packets sent by the same sender, it will be some time before it can synchronize with the schedule.

On speculation, the attack can be improved by jamming just the SYNC or CTRL interval. Consider first the case of jamming the SYNC interval and sleeping for the rest of a cycle (i.e. the CTRL interval and the sleep interval). The idea of this attack is that if a sensor node fails to receive the SYNC packets of its neighbors, it would not know it has neighbors that it can transmit data packets to. However some SYNC packets do slip through especially before the jamming motes enter their jam-sleep cycle, and when they do, the sensor nodes that receive the SYNC packets know they themselves are connected. Connected sensor nodes continue to use the CTRL interval to send data packets without problems. So jamming the SYNC interval is ineffective, as verified by simulations.

Now consider the case of jamming the CTRL interval. Jamming the CTRL interval prevents RTS/CTS handshakes, and thus data transmissions from taking place. Speculating further, instead of jamming the whole CTRL interval, we can *listen* in the CTRL interval for CTS packets, and jam the ensuing data packets instead. Since listening consumes less energy than transmission, we might be able to use less energy compared with blindly jamming the CTRL interval.

Table 1 shows the censorship rates of the 3 attacks discussed so far: jamming of the listen interval, the CTRL interval, or data packets. Figure 3 shows the attrition rates and effort ratios, with standard deviations mostly below 10%. In our simulation, the network density D ranges from 3 to 7. A network with D = 1 or 2 is too sparse to be realistic, while we have problem going beyond D = 7, being currently limited by the computational resources required to simulate networks that dense. After all, networks sparser than our simulated ones exist in practice [21]. We discuss our observation below in several aspects.

Table 1. Average censorship rates of various attacks (standard deviations in parenthesis).

(a) T_{sim} = 600s

Jammed target	Jammer count	Censorship rate R_c (%)				
		D=3	D=4	D=5	D=6	D=7
Listen interval	50	62 (25)	72 (12)	73 (28)	81 (14)	81 (16)
	100	86 (12)	90 (7)	89 (13)	87 (18)	83 (13)
CTRL interval	50	72 (19)	76 (24)	81 (16)	84 (10)	82 (26)
	100	88 (9)	93 (7)	94 (8)	94 (6)	90 (9)
Data packets	50	78 (13)	85 (11)	87 (9)	95 (3)	93 (7)
	100	96 (3)	97 (3)	97 (4)	99 (1)	99 (3)

(b) T_{sim} = 1200s

Jammed target	Jammer count	Censorship rate R_c (%)				
		D=3	D=4	D=5	D=6	D=7
Listen interval	50	58 (20)	77 (16)	77 (20)	85 (12)	88 (9)
	100	89 (7)	94 (4)	92 (8)	93 (9)	91 (7)
CTRL interval	50	70 (16)	78 (17)	84 (9)	87 (11)	87 (13)
	100	90 (6)	95 (5)	93 (10)	95 (6)	95 (4)
Data packets	50	77 (10)	84 (8)	87 (6)	94 (3)	92 (12)
	100	94 (5)	96 (3)	96 (4)	99 (1)	99 (1)

3.3.1 Network density

As expected, the lower the density of the jamming motes, or the density of the entire network, the lower the censorship rate of the motes is, since there would be more regions outside the influence of the motes. Although there are simulation runs that record 100% censorship rate, this should not in general be expected, due to the unpredictability and the dynamic nature of the simulated topology.

3.3.2 Censorship rate

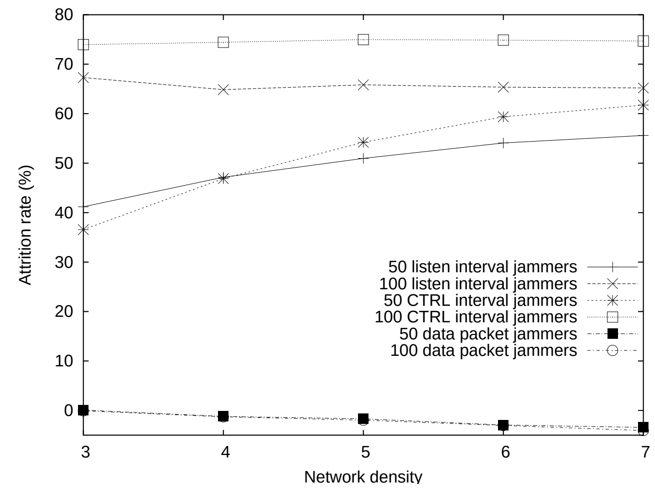
Data packet jamming has on the average the highest censorship rate for the following reason: before a jamming mote is able to synchronize with the S-MAC schedule, it can already listen for CTS packets and jam the ensuing data packets. In contrast, listen/CTRL interval jammers only listen for SYNC packets, allowing data packets to slip through, *before* starting their jam-sleep cycle. Between listen and CTRL interval jammers, CTRL interval jammers censor more data because they settle into their jam-sleep cycle earlier than listen interval jammers do.

Denote $F(t)$ as the fraction of jamming motes that still have not entered their jam-sleep cycle at time t. Figure 4 plots $F(t)$ against t, for the three types of jammers. It shows that by 200s, all CTRL interval jamming motes and all data packet jamming motes have already settled into their jam-sleep cycle, while it may take indefinitely long for all listen interval jamming motes to do so. Concluding from the preceding analysis, if we disregard the results for the first 200s (the 'settling time'), CTRL interval jammers and data packet jammers have about the same censorship rate, however listen interval jammers would still have a lower censorship rate. And because of the shorter settling time, the censorship rates measured in T_{sim} = 1200s (Table 1(b)) do not differ much from those measured in T_{sim} = 600s (Table 1(a)) for CTRL interval jammers and data packet jammers. In comparison, for listen interval jammers, the corresponding differences are larger.

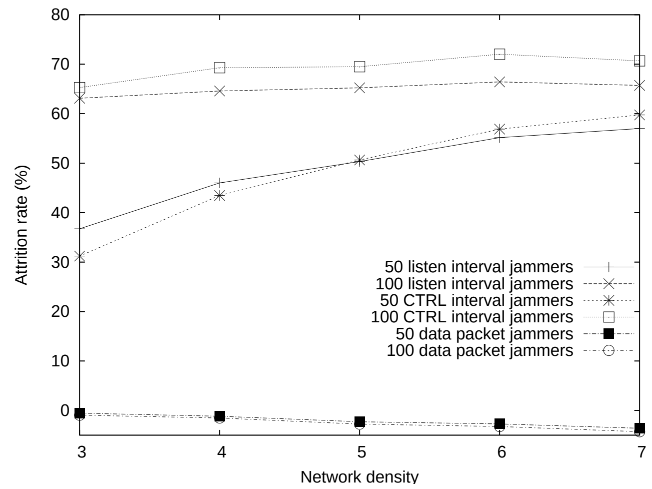
In practice, since listen/CTRL jamming relies on overlapping the jamming period with the listen/CTRL interval as closely as possible, incomplete overlapping might also allow some packets to slip through – recall in Equation 6 and 7 the use of an estimated constant δ. Note that in the controlled environment of our simulation, we are using a good estimate of δ, so this is *not* the reason for the lower censorship rate of listen/CTRL jamming in our case.

3.3.3 Attrition rate

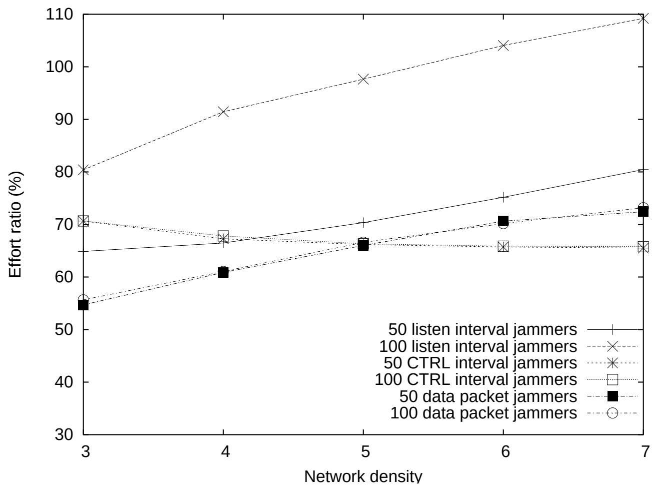
CTRL interval jamming is more successful in causing its victims to consume more energy than the other attacks are. This is because it allows its victims to listen and send SYNC packets in the SYNC interval (consuming energy for both



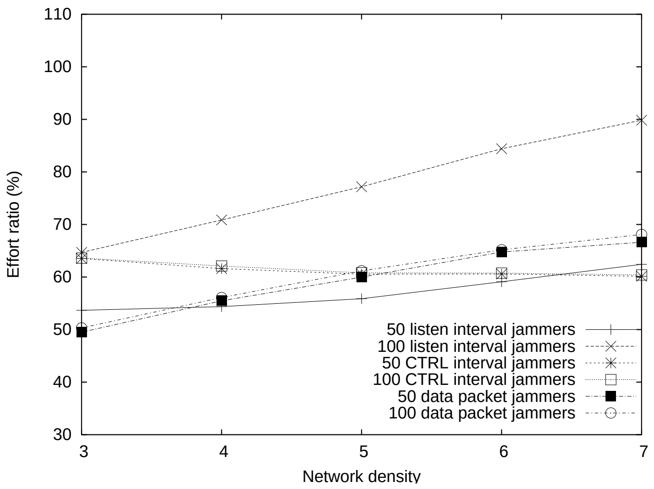
(a) Attrition rate (T_{sim} = 600s)



(b) Attrition rate (T_{sim} = 1200s)



(c) Effort ratio (T_{sim} = 600s)



(d) Effort ratio (T_{sim} = 1200s)

Figure 3. The attrition rates and effort ratios of jamming attacks.

Rx and Tx), and forces its victims to listen for the whole of the CTRL and sleep interval (consuming energy for Rx). Note that in the original implementation of S-MAC, nodes recovering from the BACKOFF state in the sleep interval (when the jamming stops) continue to listen. In comparison, listen interval jamming only forces its victims to listen for the entire period (consuming only energy for Rx). In the case of data packet jamming, sensor nodes not only waste *less* energy compared with when they are attacked by listen/CTRL interval jammers, they also waste less energy compared with when they are not under attack (negative attrition rates in Figure 3(a) and Figure 3(b)). There are two reasons:

1. The energy for retransmissions is typically less than the energy for listening for the entire sleep interval. To see this numerically, let P_{Rx} be the power consumed by listening, then the power consumed by transmitting is $(2400/960)P_{Rx} = 2.5P_{Rx}$. The energy consumed by listening for the entire sleep interval is $T_S P_{Rx}$. This is typically less than the energy consumed by sending

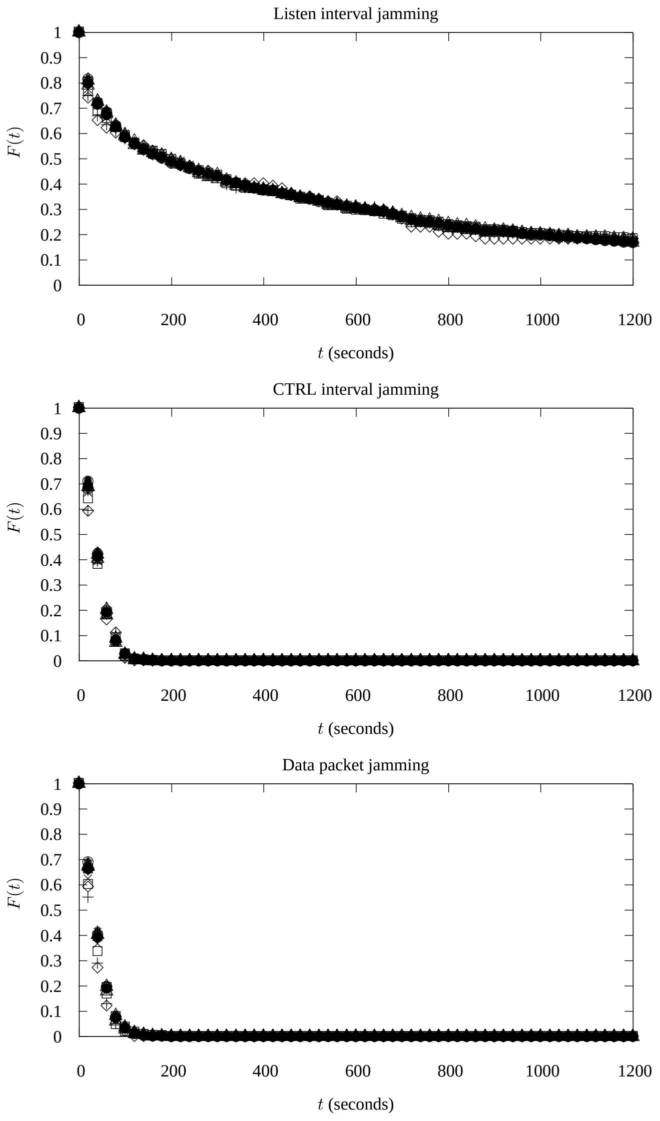


Figure 4. $F(t)$, the fraction of jamming motes that are still unsynchronized at time t, plotted against t, where number of runs = 20, number of jamming motes = 100, D = 7.

a data packet t_p seconds long which is $t_p \times 2.5P_{Rx}$, because $T_S \gg t_p \times 2.5$.

2. Sensor nodes go to sleep whenever they fail to receive acknowledgements. This uses less energy than if they continue listening (as would happen if they are attacked by listen/CTRL jammers), or transmitting (as would happen if they are not under attack and they have data to send).

It is for this reason that increasing the network density does not boost the attrition rate of 50 data packet jammers like it does for 50 listen/CTRL interval jammers, by bringing more jamming motes within the radio range of the sensor nodes (Figure 3). Notice that increasing the network density also does not boost the attrition rate of an attacker network whose density is already the same as the sensor network's (e.g. 100 jamming motes vs 100 sensor nodes), due to the saturation of the effect of attacks.

3.3.4 Effort ratio

It is not surprising that listen interval jamming has the highest effort ratio, since it transmits for the longest period of time. Notice that the effort ratio of listen interval jamming increases with network density. This is because as the network gets denser, some jamming motes are able to synchronize themselves with the schedule sooner, but as these motes start jamming the listen interval, other motes would not have the chance to capture SYNC packets, to synchronize themselves with the schedule. It is also for this very reason that the difference between 50 and 100 listen interval jammers is larger than the those between 50 and 100 CTRL interval jammers, and between 50 and 100 data packet jammers. In contrast, since CTRL interval jamming does not prevent the transmission of SYNC packets, not only are more jamming motes able to start their jam-sleep cycle sooner, but also are more jamming motes able to follow the schedule. As a result, the effort ratio of CTRL interval jamming decreases with network density. For data packet jammers, as the network gets denser, each jamming mote gets more neighbors, and hence more data packets to jam (while CTRL interval jamming motes get the same CTRL interval to jam). This explains the upward trend of the curves for data packet jammers.

It is perhaps surprising that the effort ratio of data packet jamming would exceed that of CTRL interval jamming starting from D = 5 (for both T_{sim} = 600s and 1200s) since data packet jamming is a reactive approach. One reason is that data packet jamming motes, as mentioned, already start jamming before entering their jam-sleep cycle. Another reason has to do with adaptive listening [18]. In the original S-MAC protocol [17], after a node B receives a data packet from its neighbor A, it has to wait until the CTRL interval of the next cycle, before it can relay the data packet to another neighbor C. In the improved version with adaptive listening, the node C, upon overhearing A's RTS packet, would as usual sleep when A transmits to B, but *additionally* schedule to wake up right after the transmission is over, thereby giving B the opportunity to relay the data packet to itself in the same cycle. During this adaptive listening phase, when C listens for B's potential RTS, C might also send an RTS packet if it has any data to send, thereby triggering another wave of adaptive listening. The key point is that since data packet jamming allows RTS packets to pass through, a data packet jammer needs not only to block the data packet from A to B, but also any potential data packet from C and other nodes triggered to perform adaptive listening. As long as there are RTS packets, adaptive listening may well extend deep into the sleep interval. In the case of listen/CTRL interval jamming, most RTS packets are either prevented from being sent or are corrupted by jamming signals. Nonetheless our implementation of data packet jamming has not been optimized according to Lin et al.'s error correction code-based technique [7], otherwise the effort ratio is expected to be lower.

As expected, comparing Figure 3(d) with Figure 3(c) shows that when the settling times of the jamming motes are amortized over a longer simulation period, the effort ratios become lower. The amortization effect is most pronounced with listen interval jammers as they have much longer settling times. The important point is that the increasing/decreasing trend of the effort ratios with network density remains the same for different simulation lengths.

All in all, from Equation 4, CTRL interval jamming has the highest lifetime advantage, followed by listen interval jamming, and lastly data packet jamming.

4. Countermeasure

The most straightforward solution is probably to encrypt every packet, so that jammers cannot deduce the schedule as well as differentiate the type of packets. However if a single encryption key is used for every node, then we have a single-key vulnerability. If we have more than 1 key, each node will have to send a packet encrypted with different keys each time, resulting in drastically higher energy usage. If we use just one schedule, no matter how many keys we use to encrypt the schedule, the attacker would know that schedule once it knows 1 key. If we use more than one schedule, and encrypt a different schedule with a different key, then some nodes would never learn about some schedules, resulting in some inevitable packet collisions. All in all, doing the encryption right is hard.

To counter data packet jamming which has the highest censorship rate, our proposal is to use what we call *data blurting* and *schedule switching*. The idea is that after n times timing out waiting for an ACK, the sender 'blurts' out its data packet without going through the RTS/CTS/data/ACK sequence. Upon receiving the blurted packet, the receiver announces a switch of schedule in the next cycle. After some finite number of cycles, the receiver and its neighbors would synchronize with the new schedule, which is a right shift of the old schedule along the time axis, by the length of 1 listen interval and 1 CTRL interval. We let the receiver and not the blurter announce the schedule switch because letting the blurter do it would result in too many schedule switches, which in turn would cause a vast drop in throughput. The receiver might or might not receive the blurted data packet, but once it receives a blurted data packet, it knows for sure that a schedule switch is necessary. Compare Table 1 and Table 2 to see how our technique alleviates the effect of data packet jamming, e.g. when T_{sim} = 1200s and D = 7, the censorship rate is on average reduced from 99% to 52%. Table 2 also shows that the alleviation effect gets better over time, as the jamming motes loose sync with the sensor nodes for a longer period of time. The standard deviations are large because blurted data packets might or might not be successfully received, and it takes a varying number of cycles to synchronize with a new schedule. Note that the technique does not interfere with the normal operation of the network in the absence of attacks.

Table 2. Average censorship rates of data packet jamming with countermeasure implemented (standard deviations in parenthesis).

T_{sim}	Jammer count	Censorship rate R_c (%)				
		D=3	D=4	D=5	D=6	D=7
600	50	43 (31)	50 (29)	57 (27)	64 (31)	61 (28)
	100	64 (21)	55 (20)	66 (29)	68 (32)	72 (23)
1200	50	35 (26)	34 (26)	27 (38)	54 (32)	47 (33)
	100	50 (18)	40 (29)	50 (22)	56 (36)	52 (49)

5. Related work

Link-layer attacks are not new but are certainly not actively studied either. Wood et al. acknowledged in 2002 that no effective defense was yet known against link-layer jamming [16]. Both Ståhlberg [14] and Wood et al. [16] quote how an attacker might keep sending RTS packets to elicit CTS packets from its victims, as an example of *sleep deprivation torture attacks* [13]. We think the attack is irrelevant in the case of S-MAC where nodes go to sleep periodically – it would be long before the energy of the sensor nodes is exhausted, and in contrast it would not be too long before the attacker exhausts its own energy first.

Negi and Perrig [9] have investigated 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. In reality, RTS packets are typically too short to react to in time, jamming the ensuing CTS or data packets is more feasible. The attacker also has to send RTS packets that can pass the integrity check by the normal sensor nodes. Our attackers are not bound by such an assumption.

Konorski [4] has proposed a scheduling policy that addresses the selfish behavior of using small or no contention windows in contention-based protocols, in the context of *single-hop* networks. We are only interested in multi-hop networks. Kyasanur and Vaidya [5] targets selfishness in IEEE 802.11, specifically in the random backoff mechanism. Čagalj et al. [15] 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.

6. Conclusion

Among all attacks on network protocols of sensor networks, we argue that link-layer attacks are the most relevant, on the grounds of energy efficiency, and given the fact that the correct functioning of any higher-level protocol depends on a working link layer. In this paper, we have demonstrated both qualitatively and quantitatively how listen interval jamming, CTRL interval jamming and data packet jamming affect S-MAC. Based on our results, we have shown that data packet jamming is better than the other attacks in censorship rate by a small margin. We have also shown that CTRL interval jamming has the best lifetime advantage. As a countermeasure to data packet jamming, we have proposed a new technique based on data blurting and schedule switching. We realize that we need a better approach to reduce the standard deviations of this technique, and also a countermeasure to the other attacks. At the same time, we also plan to report more attacks on S-MAC, particularly in how the attackers might analyze encrypted S-MAC traffic and respond to changes in the schedule, and validation of our simulations in a later paper.

References

[1] 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 DEST International Workshop on Signal Processing for Sensor Networks*. IEEE Computer Society Press, 2004.

[2] IEEE. IEEE Standards for Information Technology – Telecommunications and Information Exchange between Systems – Local and Metropolitan Area Network – Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Nov. 1999.

[3] 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.

[4] 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.

[5] 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.

[6] Y. Law, J. Doumen, and P. Hartel. Benchmarking block ciphers for wireless sensor networks (extended abstract). In *1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2004)*. IEEE Computer Society Press, Oct. 2004.

[7] G. Lin and G. Noubir. Low Power DOS Attacks in Data Wireless LANs and Countermeasures. Technical report, Northeastern University, 2002.

[8] M. Mysore, M. Golan, E. Osterweil, D. Estrin, and M. Rahimi. TinyDiffusion in the Extensible Sensing System at the James Reserve. Web page, May 2003.

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

[10] T. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, 2nd edition, 1996.

[11] 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)*. IEEE Computer Society Press, 2004.

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

[13] F. Stajano and R. Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In *Proceedings of The 7th International Workshop on Security Protocols*, volume 1796 of *LNCS*, pages 172–194. Springer-Verlag, 2000.

[14] 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, Fall 2000.

[15] M. Čagalj, S. Ganeriwal, I. Aad, and J.-P. Hubaux. On Cheating in CSMA/CA Ad Hoc Networks. Technical Report IC/2004/27, EPFL-IC, 2004.

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

[17] 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.

[18] 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.

[19] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *Proc. IEEE INFOCOM*, volume 2, pages 1312–1321, Apr. 2003.

[20] 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.

[21] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi. Hardware design experiences in ZebraNet. In *2nd International Conference on Embedded Networked Sensor Systems*, pages 227–238. ACM Press, 2004.