Precise Packet Loss Pattern Generation by Intentional Interference

Zhitao He, Thiemo Voigt
Swedish Institute of Computer Science
Box 1263, SE-164 29
Kista, Sweden
Email: {zhitao, thiemo}@sics.se

***Abstract*—Intermediate-quality links often cause vulnerable connectivity in wireless sensor networks, but packet losses caused by such volatile links are not easy to trace. In order to equip link layer protocol designers with a reliable test and debugging tool, we develop a reactive interferer to generate packet loss patterns precisely. By using intentional interference to emulate parameterized lossy links with very low intrusiveness, our tool facilitates both robustness evaluation of protocols and flaw detection in protocol implementation.**

I. Introduction

Robustness of a low-power wireless sensor network can be undermined by poor channel conditions, plagued by unpredictable noise and interference. Empirical studies investigating the relation between packet losses and channel parameters have revealed volatile behaviors of intermediate quality links in realistic channel conditions [1], [2], [3]. The impact of volatile links to network connectivity can be evaluated in a network simulator augmented by analytical or statistical packet loss models, as shown by Lee et al. [4]. A more recent empirical research by Srinivasan et al. focuses on measuring individual links' burstiness in a testbed [5]. They demonstrate that relaxing link-layer retransmission time-outs can mitigate bursty link failures, which in turn improves end-to-end packet delivery rates over a multi-hop network. Link failures can be created with intentional radio interference, which allows the user to study network operations under certain controlled conditions. Boano et al. [6] has developed an interferer based on an IEEE 802.15.4 radio transmitter, which can reliably block a channel for arbitrary time intervals by injecting a strong jamming signal. They show that duty-cycling the interferer can yield precise packet loss rate (PLR), when the interferer is configured with large blocking time intervals that essentially slices the channel into on-off time slots. This would however result in unrealistically high link burstiness, causing severe fluctuations in network performance, thus limiting the user's ability to investigate many intricacies caused by intermediate-quality links. Furthermore, because most mechanisms for recovering packet losses are implemented by the link layer or the network layer, the ability to construct different sequences of packet losses is often more useful than the ability to control variations on the physical channel.

We have conceived and developed a reactive interferer that enables arbitrary packet loss pattern generation based on packet detection. Leveraging the half-duplex property of common sensor radios, our reactive interferer follows a listen-before-send principle, locking to a packet flying over the air and then destroying it before it reaches its intended destination. Given a sufficient link margin, near-perfect detection and destruction rates can be attained, guaranteed by reliable header detection by modern digital transceivers such as CC2420. This leaves the control of packet loss rate and burstiness lengths completely at the mercy of a jamming decision function defined by the user, a flexibility usually only available with software-based simulators. An additional merit of our interferer is its low intrusiveness to the rest of the network, as it only transmits for a short interval when a packet is detected.

We demonstrate how the interferer can be used to generate different loss patterns. The high precision and programmability of our tool enable a protocol designer to evaluate protocol robustness against various interference sources. Particularly, header decoding makes selective blocking possible, allowing the user to emulate rare link statuses such as asymmetric bidirectional links. Furthermore, we show how protocol design flaws can be uncovered by using the interferer to explore protocol states.

The rest of the paper is organized as follows. Section II discusses related work; Section III and IV show our design and implementation of the interferer; We show the use of our tool in Section V. We discuss the limitation of the tool in Section VI and draw our conclusions in Section VII.

II. Related Work

A number of previous work studied jamming attacks launched by normal sensor radios, a malicious form of intentional interference, from a defender's perspective. Law et al.'s link layer jamming analysis showed that collection of packet timing statistics allowed an intelligent jammer to infer the timing specification of a MAC protocol to carry out energy-efficient attacks [7], [8]. Xu et al. compared four MICA2-based jammers, including a reactive one based-on signal strength sensing, and showed that the reactive jammer posed a particularly high threat for its high blocking rate and small energy footprint [9]. Wood et al. studied the threats by a number of IEEE 802.15.4 PHY layer jamming attacks, including our SFD detection-based mechanism, and developed respective defense measures [10].

Boano et al. invented a convenient interference generator based on the CC2420 radio's continuous transmission mode, shedding light on link behaviors at the presence of a strong, local interference [6]. Exploiting the same hardware technique, JamLab [11] further enables the user to approximately reproduce the power trace recorded from real interference sources, which facilitates repeatable testbed experiments. Reconstruction of the "crime scenario" requires fast, high-resolution spectrum scanning and precise transmission power control, which JamLab achieves by utilizing the CC2420 radio transceiver's maximum raw capacity and applying data compression to power samples. Our reactive interferer enables the user to manipulate individual packet receptions to control a specific link's loss pattern, which can be used to answer hypothetical questions related to robustness design of protocols. Operating on an abstraction level above JamLab, our interferer does not require intensive signal processing, and it is less intrusive to the rest of the network than an active interferer.

III. Design

Our interferer is usually placed within the mutual transmission range of two communicating nodes, to ensure a sufficiently large link budget for reliable detection and destruction of packets that travel over the link. It operates in sniffer mode most of the time, listening to the channel; when a packet is detected, it queries a jamming decision function which determines whether to intervene the ongoing communication. Figure 1 illustrates the sniffer and interferer modes.

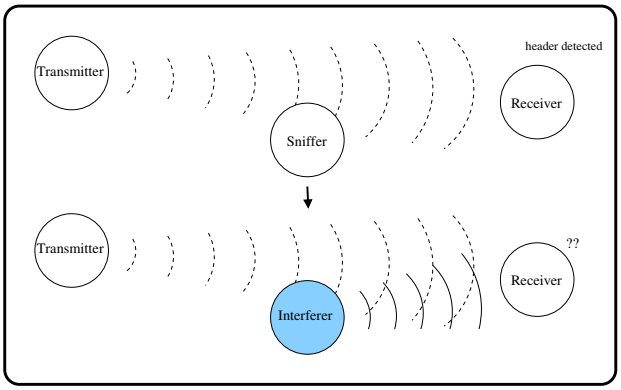


Fig. 1. The interferer first sniffs a packet, before intervening its reception by sending a jamming signal.

A. SFD Sniffer

For incoherent digital receivers, frame synchronization usually relies on detection of a synchronization header agreed with the transmitter before hand. For both IEEE 802.11b and IEEE 802.15.4, the synchronization header consists of a preamble sequence followed by a fixed start-of-frame delimiter (SFD). SFD detection is the result of correct symbol demodulation, thus is more reliable than carrier sensing methods purely based on received signal strength. For an IEEE 802.15.4-compliant receiver such as CC2420, automatic detection of the preamble sequence and the SFD (4 zero bytes followed by 0x7A) triggers the start of a packet reception procedure that writes the rest of the decoded data into an RX FIFO. When two receivers detect the same packet, both start this process synchronously. The lowest time-line of Figure 2 shows how a second, unintended receiver, in this case our interferer, sniffs a incoming packet after successful synchronization with the transmitter's preambles and SFD. A necessary condition for successful detection is that the received signal strength is above the sensitivity of the interferer (-95 dBm for CC2420). Figure 2 shows the timing of the interference generation.

B. Interference Transmission

Generation of interference from a half-duplex radio requires the interferer to switch from receiving state into transmission state. For CC2420, the RX-TX switch time is determined by the settling time of the frequency synthesizer. The transmission of the interference signal only needs to overlap a portion of the packet payload in order to cause a checksum error at the receiver.

In principle, successful destruction of the target packet requires a sufficiently strong interference signal to degrade the receiver's (SNIR) below its co-channel rejection threshold (-3 dB for CC2420). In reality, due to a certain degree of delay capture effect, the interference signal needs a small extra power margin, compared with the original transmitter, in order to destroy the reception reliably.

C. Jamming Decision Function

Our interferer features a user customizable jamming decision function, which can selectively bypass certain detected packets. The function can be used to implement an arbitrary kind of loss model, turning a perfectly connected link into a partially connected link characterized by certain loss probability and burstiness. We demonstrate the use of a number of jamming decision functions in Section V.

D. Optional Header Decoding

In additional to SFD detection, our sniffer can choose to further decode the following PHY header, which provides more options for the user to be selective about which packets to block. For IEEE 802.15.4, the PHY header consists of an 8-bit packet length field. The interferer may even be used to decode the following MAC header to intercept address information, which enable more complex manipulation.

IV. Implementation

The sniffing and interference generation functions of our reactive interferer can be directly implemented as an interrupt service routine (ISR) inside the radio driver. We have chosen to add these additional functions to the Contiki operating system's CC2420 radio driver. To enable fast packet detection, we set CC2420s clear channel assessment (CCA) to the SFD mode, by configuring CC2420s *MDMCTRL0* register to *CCA_MODE 2*. Meanwhile, we enable the MCUs interrupt vector for the CCA pin, so that a distinct hardware interrupt

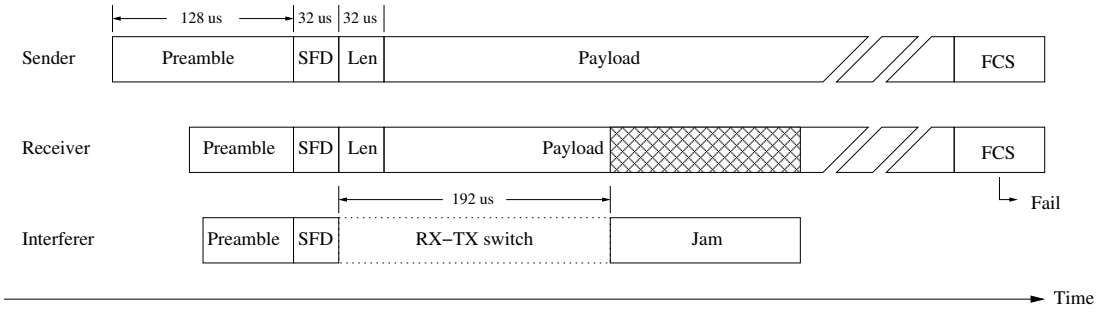


Fig. 2. Interferer sniffs the channel to synchronize with any packet using SFD detection, and then switches into transmission mode to send a short blocking signal to corrupt the payload, which forces the intended packet receiver(s) to drop the packet.

TABLE I
KEY PARAMETERS OF THE REACTIVE INTERFERER

Minimum PHY payload length of target packet	3 bytes
Sender-interferer delay	352 µs
Minimum effective jam burst length	312 µs
Interferer TX power margin over sender	5 dB

is raised as soon as an SFD is detected. The interferer can be reversed to a normal receiver by disabling this special CCA interrupt.

Within the CCA ISR, the user can optionally wait to retrieve extra data from the RX FIFO, before passing them to the jamming decision function. If the function returns true, the radio switches to TX state to transmit a continuous jamming signal generated using CC2420's modulated carrier transmission mode. We essentially reuse the power-adjustable jammer by Boano et al. [6], but only enable the transmission for a burst period of about 312 µs, so as to minimize both channel occupation and processing delay. The waveform of the jam signal is modulated carrier. To prepare for the next sniff, we flush the RX FIFO before re-enabling the CCA interrupt.

Because of its simplicity, the overhead in terms of code and memory space of the interferer is negligible, and the processing delay in largely determined by the complexity jamming decision function.

We summarize the key parameters of the reactive interferer in Table I.

V. Evaluation

Our evaluation experiments consist of a set-up of 3 Tmote Sky nodes: a sender, a receiver and an interferer, all placed within mutual transmission ranges. When the interferer is in sniffer mode, the sender-receiver pair sees a perfectly connected link; when the interferer is in transmission mode, the link is blocked completely, due to the interferer's high power margin over the transmitter. We first showcase how a set of basic jamming decision functions can yield partial unidirectional links with precisely-controlled packet loss rates (PLR). We then further show how to manipulate partial failures on a bidirectional link that runs a reliable link layer retransmission protocol, which enables us to explore all protocol states and evaluate the protocol's robustness.

A. Unidirectional Link

In this experiment we show that it is possible construct a lossy link between a transmitter-receiver pair, and that its loss pattern can be precisely controlled by applying different jamming decision functions to block portions of the transmissions. The transmitter sends a sequence of packets to the receiver, each embedding a unique sequence number; the receiver prints out the seqno. of received packets, whereby a missing seqno. identifies a unreceived packet.

For convenience, we use the *identified broadcast* primitive provided by Contiki's Rime stack to implement a unidirectional link [12]. The packet sequence consists of frames of 50-byte sizes, sent at 128 Hz rate. The interferer is configured to react on SFD detection without additional header decoding. Arbitrary PLR can be realized by insertion of a jamming decision function between the detection and interference transmission function to "filter out" specific packets. To demonstrate the potential diversity in achievable loss patterns, we implement 3 different jamming decision functions, all emulating a link with a 25% PLR but suffering different degrees of bursty losses. The first function emulates a strong interference source emitting short, periodic power spikes, such as a WiFi access point, which causes a one loss in every four packets. The second function emulates a similar interference source, which emits at double intensity but at half frequency, resulting bursty losses of two packets in a row. The third function uses a pseudo random number generator to model a link as a Bernoulli process with a 0.25 loss probability for each individual packet. Figure 3 shows the receiver's observed link connectivity status over a period of 160 packets, when exposed to the three emulated interference sources. Even under a high packet rate of 128 pks/s, our interferer only requires an average channel duty cycle of 1% to incur 25% losses. An active interferer would need to transmit on 25% duty-cycle in order to achieve the same PLR, regardless of the actual packet rate. But a high duty-cycle transmitter potentially violates regional ISM band regulations.

B. Bidirectional Link

Data delivery over a multihop sensor network usually requires a link layer unicast component to relay data packets through each hop on the path. To enhance robustness against

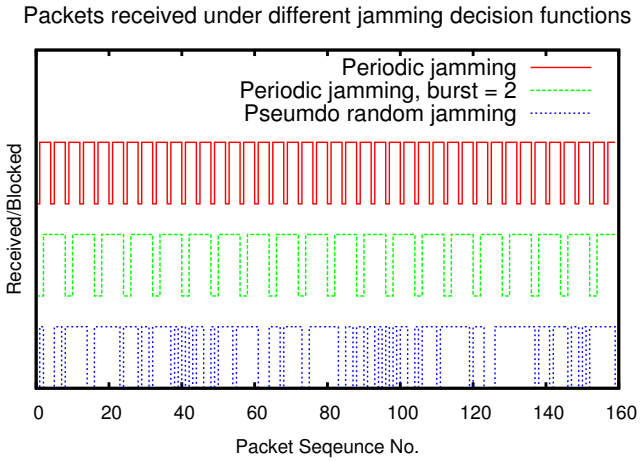
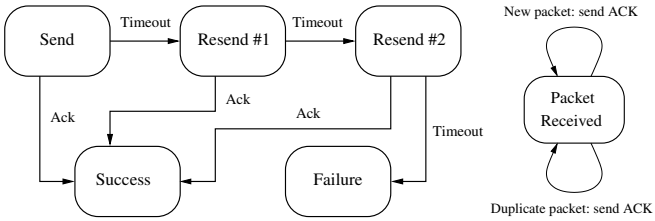


Fig. 3. Three lossy unidirectional links between by the same transmitter-receiver pair, whose loss patterns are emulated by different jamming decision functions.



(a) Transmitter (b) Receiver

Fig. 4. State transition diagrams for Rime runicast

occasional link failures due to local channel fluctuations, a packet is usually buffered by an intermediate node and retransmitted to its next-hop destination when an expected acknowledgment packet from the latter does not arrive. Exchange of pairs of data and acknowledgment packets forms a traffic pattern, where a communication failure in either packet results in a further retransmission. Retransmissions of a data packet usually repeat a number of times up to a maximum count, then the packet is dropped. Depending on when a failure occurs on the bidirectional link, this recovery mechanism can traverse along different paths of an internal protocol state machine. We want to demonstrate how a protocol designer can use the interferer to explore such state traversals for debugging purposes.

Contiki Rime's *reliable unicast (runicast)* module provides a standard link layer retransmission scheme, with configurable retransmission intervals and maximum retransmission count. A runicast acknowledgment packet consists of 18 bytes, including packet type, sequence number and address information. We run a runicast application between a transmitter-receiver pair, where the transmitter sends data packets at 10 second intervals, and up to two retransmissions are scheduled, one second apart. Figure 4a and Figure 4b show the transmitter and the receiver's state diagrams respectively. Note that the receiver distinguishes a duplicate data packet by comparing its sequence number with the previous packet. We configure our interferer to decode the IEEE 802.15.4 PHY header, to reveal the length of the packet being intervened.

1) Symmetric link: We emulate a symmetric bidirectional link, where both forward and reverse links are lossy. The interferer uses a Bernoulli jamming decision function to block 20% of all detected packets. We show in Figure 5 a trace that indicate the packets sniffed by the interfere as well as the retransmission count observed from the transmitter over a period of 300 seconds. Data packets and their respective acknowledgments are separated by 10-s intervals. In additional to this pattern, some data packets are clustered with one-second intervals, indicating repeated transmissions of the same data. Because loss can occur on both directions, retransmission can be caused by a lost data packet or a lost acknowledgment. Throughout the period, all packets make to the receiver after at most two retransmission attempts.

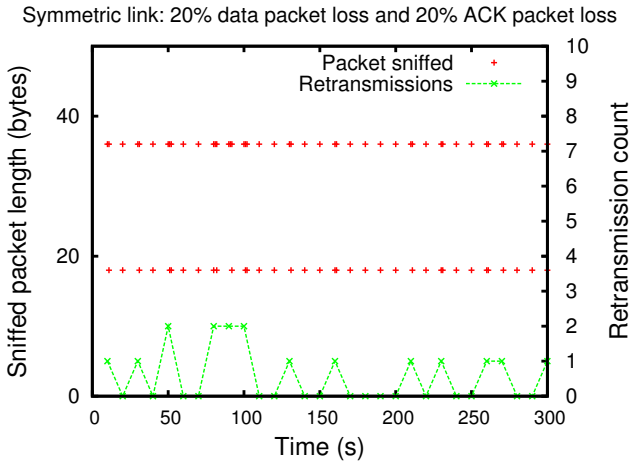


Fig. 5. An emulated symmetric link with loss rate of 20% on both directions

2) Asymmetric Link With Forward Loss: We emulate an asymmetric bidirectional link, where the forward link is lossy but the reverse link is perfect. The interferer identifies the data packets sent via the forward link by PHY header decoding, and then uses a Bernoulli jamming decision function to block 25% of those packets; the acknowledgment packets sent via the reverse link are left intact. We show in Figure 6 a trace that show the packets sniffed and retransmission count over a period of 300 seconds. At the time 260 s, all the three transmission attempts fail to reach the receiver, thus resulting in no acknowledgment packet sent.

3) Asymmetric Link With Reverse Loss: We emulate another asymmetric bidirectional link, where the forward link is perfect but the reverse link is lossy. We configure the interferer to cause 40% of acknowledgment packets to get lost. In the trace shown in Figure 7, lost ACK packets cause additional retransmissions of the buffered data packet.

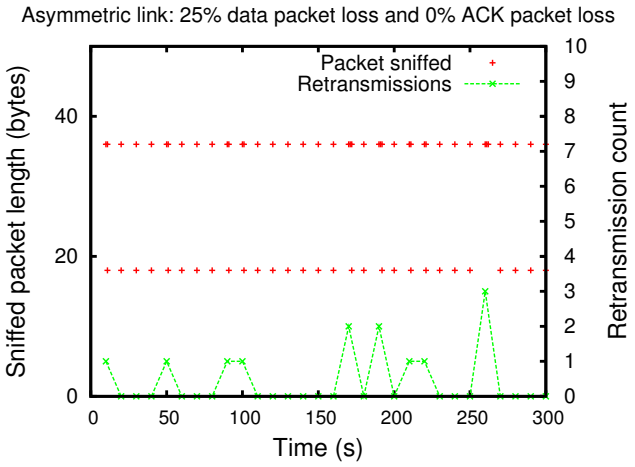


Fig. 6. An emulated asymmetric link with 25% forward loss

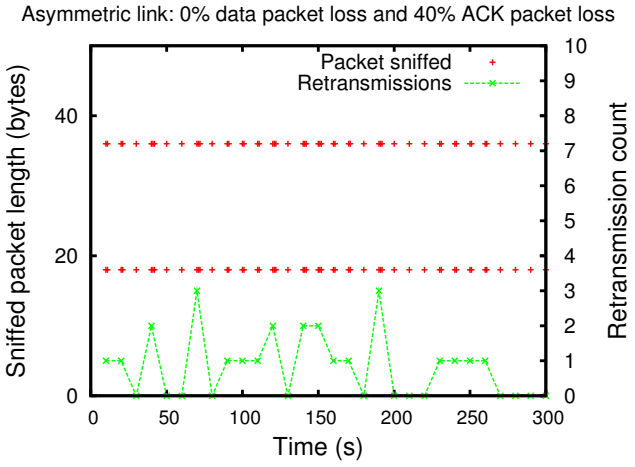


Fig. 7. An emulated asymmetric link with 40% reverse loss.

4) Debugging Protocol Flaws: The runicast module contains a relatively simple state machine, which we are able to traverse fully in short time. To our surprise, despite the simplicity of the protocol logic, we discovered two programming errors in the implementation. When we experimented with a fully blocked reverse link, the sniffer observed excessive transmissions of packets after the maximum retransmission count was reached. After investigating the code, we found the problem was caused by redundant triggering of the retransmission timer, one from the runicast module and the other from the underlying stubborn unicast module. This "harmless" bug did not affect the overall function, and had escaped the nightly automatic tests in the Contiki simulator, because the simulator only verified positive messages from the top application layer.

After fixing the excessive retransmissions, we discovered that while only the reverse link was blocked, the receiver could receive all new data packets from the forward link, but mistook them as old redundant packets and discarding them. It turned out that the problem had arisen from the transmitter's state machine. Under normal conditions, the transmitter maintains a packet sequence number and increments it upon every received acknowledgment. This is done in a callback function triggered by the received ACK packet. When a packet fails all retransmissions because no ACKs are detected, the incrementation should instead occur in the "time out" callback. In our case this other increment statement was neglected by the programmer, probably because that code branch had been visited rather infrequently. The result was that while the ACKs were blocked the transmitter kept transmitting the same sequence number for new data packets. Only until an ACK was received then the transmitter's state machine would switch out of the faulty code branch. Since such a persistent disconnection of the reverse link is rare in a dense testbed, this bug appears to have a "self-healing" property: when link layer retransmissions of a data packet occasionally fails, it incurs discarding of the next data packet as well, resulting in a bursty loss of two packets at the network layer, but no worse consequences. Under regression tests that only measure average end-to-end data delivery rates, occasional bursty disconnections due to such a flaw at the link layer is very unlikely to be detected. Our tool has shown its capability to assist state machine debugging at the lower protocol layers.

VI. Limitations

An important limitation of the interferer is its minimum RX-TX switching time requirement of 192 us, which correponds to 6 data bytes. A radio transceiver with faster switching time would allow us to block short packets such as the IEEE 802.15.4 ACK packets and MAC layer strobes used by protocols such as X-MAC [13]. Moreover, concurrent packet transmissions would be missed by the interferer. Another limiting factor is the effective detection range and effective inteference range of the transceiver, a compliant radio front-end module would improve the sensitivity and boost the transmission power, thus increasing both ranges.

VII. Conclusion

We have developed a reactive interferer to assist WSN protocol testing and debugging, by emulating different packet loss patterns with high precision. We show that not only robustness can be evaluated by applying fine-controlled, intentional interference, tricky implementation flaws in a link protocols can be more easily uncovered using the tool.

REFERENCES

[1] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, (Los Angeles, California, USA), pp. 1–13, ACM, 2003.

[2] M. Zuniga and B. Krishnamachari, "Analyzing the transitional region in low power wireless links," in *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, pp. 517–526, 2004.

[3] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, "Impact of radio irregularity on wireless sensor networks," in *Proceedings of The International Conference on Mobile Systems, Applications, and Services (MobiSys)*, (Boston, MA, USA), pp. 125–138, ACM, 2004.

[4] H. Lee, A. Cerpa, and P. Levis, "Improving wireless simulation through noise modeling," in *Proceedings of the 6th international conference on Information processing in sensor networks*, pp. 21–30, ACM, 2007.

[5] K. Srinivasan, M. Kazandjieva, S. Agarwal, and P. Levis, "The β-factor: measuring wireless link burstiness," in *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, (Raleigh, NC, USA), 2008.

[6] C. A. Boano, Z. He, Y. Li, T. Voigt, M. Zuniga, and A. Willig, "Controllable Radio Interference for Experimental and Testing Purposes in Wireless Sensor Networks," in *Proceedings of the 4th International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, (Zurich, Switzerland), Oct. 2009.

[7] Y. Law, P. Hartel, J. den Hartog, and P. Havinga, "Link-layer jamming attacks on S-MAC," in *Proceeedings of the Second European Workshop on Wireless Sensor Networks*, pp. 217–225, 2005.

[8] Y. W. Law, M. Palaniswami, L. V. Hoesel, J. Doumen, P. Hartel, and P. Havinga, "Energy-efficient link-layer jamming attacks against wireless sensor network mac protocols," *ACM Trans. Sen. Netw.*, vol. 5, no. 1, pp. 1–38, 2009.

[9] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '05, (New York, NY, USA), pp. 46–57, ACM, 2005.

[10] A. Wood, J. Stankovic, and G. Zhou, "DEEJAM: Defeating Energy Efficient Jamming in IEEE 802.15.4-based Wireless Networks," in *The IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (IEEE SECON)*, June 2007.

[11] C. Boano, T. Voigt, C. Noda, K. Römer, and M. Zúñiga, "JamLab: Augmenting Sensornet Testbeds with Realistic and Controlled Interference Generation," in *Proceedings of the 10th international conference on information processing in sensor networks (IPSN)*, 2011.

[12] A. Dunkels, F. Österlind, and Z. He, "An adaptive communication architecture for wireless sensor networks," in *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, (Sydney, Australia), Nov. 2007.

[13] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks," in *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, (Boulder, Colorado, USA), 2006.