Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2010, Article ID 819318, 16 pages
doi:10.1155/2010/819318

Research Article

Spatio-Temporal Techniques for Anti-Jamming in Embedded Wireless Networks

Miroslav Pajic and Rahul Mangharam

Department of Electrical and Systems Engineering, University of Pennsylvania, PA 19104-6243, USA

Correspondence should be addressed to Rahul Mangharam, rahulm@seas.upenn.edu

Received 12 November 2009; Accepted 10 February 2010

Academic Editor: Ingrid Moerman

Electromagnetic jamming results in a loss of link reliability, increased energy consumption and packet delays. In the context of energy-constrained wireless networks, nodes are scheduled to maximize the common sleep duration and coordinate communication to extend their battery life. This coordination results in statistical and predictable activity patterns that may be easily detected and jammed. To eliminate spatio-temporal patterns of communication in the link and network layers, we present WisperNet, an energy-efficient anti-jamming protocol. WisperNet employs hardware-based time synchronization and lightweight cryptographic hashing for coordinated temporal randomization of slot schedules at the link layer and adapts routes to avoid static jammers in the network layer. We demonstrate that WisperNet reduces the efficiency of any statistical jammer to that of a random jammer, which has the lowest censorship-to-link utilization ratio. In the presence of a statistical jammer, WisperNet provides sub-2% packet drop ratios for link utilization up to 50%. In addition, the jammer's censorship efficiency is linear with link utilization as it is unable to extract any communication patterns. WisperNet is more efficient than low-power CSMA protocols in terms of energy, effective network throughput, reliability and delay.

1. Introduction

Resilience to electromagnetic jamming and its avoidance are difficult problems. It is often both hard to distinguish malicious jamming from congestion in the broadcast regime and a challenge to conceal the activity patterns of the legitimate communication protocol from the jammer. Jamming may be both malicious with the intention to block communication of an adversary or nonmalicious in the form of unintended channel interference. In the context of embedded wireless networks for time-critical and safety critical operation as in medical devices and industrial control networks, it is essential that mechanisms for resilience to jamming are native to the communication protocol. Resilience to jamming and its avoidance, collectively termed as *anti-jamming*, are a hard practical problem as the jammer has an unfair advantage in detecting legitimate communication activity due to the broadcast nature of the channel. The jammer can then emit a sequence of electromagnetic pulses to raise the noise floor and disrupt communication. Communication nodes are unable to differentiate jamming signals from legitimate transmissions or changes in communication activity due to node movement or nodes powering off without some minimum processing at the expense of local and network resources.

In the case of energy-constrained wireless sensor networks, nodes are scheduled to maximize the common sleep duration and coordinate communication to extend their battery life. With greater network synchronization, the communication is more energy-efficient as nodes wake up from low-power operation just before the common communication interval. Such coordination introduces temporal patterns in communication with predictable intervals of transmission activity. Channel access patterns make it efficient for a jammer to scan and jam the channel only during activity intervals. The jammer can time its pulse transmission to coincide with the preambles of packets from legitimate nodes and thus have a high censorship to channel utilization ratio while remaining difficult to detect. The jammer is thus able to exploit the temporal patterns in

communication to disrupt a transmission of longer length of legitimate transmissions with a small set of jamming pulses.

For nodes in fixed locations, a jammer can select regions with heavier communication activity or denser connectivity to increase the probability that a random jamming pulse results in corrupting an on-going transmission. Nodes in the proximity of the jammer will endure a high cost of operation in terms of energy consumption and channel utilization with a low message delivery rate. They must either physically relocate or increase the cost of their links so the network may adapt its routes.

Methods for antijamming must therefore address threats due to both temporal patterns at the link layer and spatial distribution of routes in the network layer. Our goal is to reduce or eliminate spatiotemporal patterns in communication while maintaining energy-efficient, coordinated and collision-free operation in multihop wireless sensor networks. We achieve this by incorporating coordinated temporal randomization for slot schedules and slot durations between each node and its k-hop neighbors. This prevents the jammer from predicting the epoch and length of the next activity on the channel. Such mechanisms reduce the effectiveness of any statistical jammer to that of a random pulse jammer. While temporal randomization prevents statistical jammers from determining any useful packet interarrival distribution for preemptive attacks, it still has an efficiency of a random jammer and can achieve censorship which increases linearly with channel utilization and jamming activity. To avoid such random jammers which are colocated near nodes with active routes, we employ adaptive routing to select paths such that the highest possible end-to-end packet delivery ratios are achieved. We combine the above temporal and spatial schemes in a tightly synchronized protocol where legitimate nodes are implicitly coordinated network-wide while ensuring no spatiotemporal patterns in communication are exposed to external observers.

In the context of multihop embedded wireless networks, which are battery-operated and require low-energy consumption, we require the following properties from the antijamming protocol.

(1) Nonpredictable Schedules. Transmission instances (e.g., slot assignments) are randomized and nonrepeating to prevent the jammer from predicting the timing of the next slot based on observations of channel activity. In this way, even if the jammer successfully estimates slot sizes, it has to transmit pulse attacks at an interval of the average slot duration to corrupt communication between nodes. This makes it especially inefficient for the jammer operating on networks with low-channel utilization, which is the common case in sensor networks. In this case, the jammer must expend several times more energy in jamming pulses to corrupt a single legitimate transmission.

(2) Nonpredictable Slot Sizes. Slots are randomly sized on a packet-by-packet basis in order to prevent the jammer from estimating the duration of channel activity for energy efficient reactive jamming. This requirement further reduces the jammer's lifetime as it will need to employ the smallest observed slot duration as its jamming interval.

(3) Coordinated and Scheduled Transmission. The communication schedule according to which a node transmits is known to all of its legitimate neighbors so they can wake up to receive the message during its transmission slot. This also prevents nodes from turning on their receiver when no legitimate activity is scheduled and hence reduces the likelihood of a jammer draining the energy of a node. The schedule must be determined implicitly without the need for frequent and explicit node-to-node control message exchange. This allows energy-efficient operation and increases lifetime of every node in the network.

(4) Coordinated Changes of Slot Sizes. All nodes must be aware of the current and next slot sizes. This is very important because any incompatibility or synchronization error would disable communication between legitimate nodes.

(5) Collision-Free Transmission. Communication must satisfy the hidden terminal problem so that a transmit slot of a given node does not conflict with transmit slots of nodes within its k-hop interference range. For scheduled communication with randomized slot assignment, it is essential that scheduling conflicts are eliminated implicitly and incur no additional overhead.

The rest of this paper is organized as follows. In Section 2, we provide a background and related work for energy efficient protocols and energy efficient jamming schemes. In Section 3, we provide an overview of the WisperNet antijamming protocol and describe the coordinated temporal randomization scheme. In Section 4, we describe the WisperNet coordinated spatial adaptation scheme. Section 5 describes our implementation experiences and experimental results followed by the conclusion.

This paper is an extended version of [1]. In this paper, we present further details about all aspects of the WisperNet. An improved algorithm for implicit conflict resolution is evaluated. A redesigned algorithm for topology maintenance and updates with conflict-free operation replaces the previous contention-based scheme. In addition a more integrated approach for implementing WisperNet has been described. WisperNet has been implemented in the application layer on top of the existing real-time operating system. This allows a straightforward method for existing applications to use WisperNet. Finally, we demonstrate the low-memory requirements of the WisperNet protocol.

2. Background and Related Work

To understand the inherent trade-off between energy efficient link protocols with well-defined schedules and their susceptibility to jamming attacks, we first describe the different types of jammers and their impact on various types of link layer protocols. We then highlight a particular class of statistical jammers and their impact on energy-efficient sensor network link protocols.

2.1. Jammers and Trade-Offs with Jamming

2.1.1. Comparison of Jamming Models. In [2, 3], Xu et al. introduce four common types of jammers: constant, random, reactive, and deceptive. Constant jammers continually emit a jamming signal and achieve the highest censorship of packets corrupted to total packets transmitted. The constant jammer, however, is not energy-efficient and can be easily detected and localized. The random jammer is similar to the constant jammer but operates at a lower duty cycle with intervals of sleep. A random jammer transmits a jamming signal at instances derived from a uniform distribution with known minimum and maximum intervals. The censorship ratio of the random jammer is constant and invariant to channel utilization. At low duty cycles, the random jammer is difficult to detect and avoid. A reactive jammer keeps its receiver always on and listens for channel activity. If a known preamble pattern is detected, the reactive jammer quickly emits a jamming signal to corrupt the current transmission. Reactive jammers, while effective in corrupting a large proportion of legitimate packets, are not energy efficient as the receiver is always on.

Another type of reactive jammer uses a simple physical layer energy detector as sensing and wake-up radios. These agile jammers wait until channel activity is detected and then jam. Although energy to "listen" is lower, this behavior is also energy inefficient since any kind of channel activity triggers a transmission of a jamming pulse. Due to physical layer delays, these jammers are effective in jamming the fraction of packets that are greater than a certain threshold length.

A deceptive or protocol-aware jammer is one that has knowledge of the link protocol being used and the dependencies between packet types. Such a jammer exploits temporal and sequential patterns of the protocol and is very effective.

In [4], a statistical jamming model is described where the jammer first observes temporal patterns in channel activity, extracts a histogram of interarrival times between transmissions, and schedules jamming pulses based on the observed distribution. This results in a very effective jammer that is not protocol-aware and is also difficult to detect. A statistical jammer chooses its transmission interval to coincide with the peak of interarrival times and is thus able to maximize its censorship ratio with relatively little effort. Figure 1(a) illustrates the relative censorship ratio and the energy-efficiency of the different jammers. Figure 1(b) illustrates the relative stealth or difficulty in detection. We observe that the statistical jammer has a high censorship ratio with both energy-efficient and stealthy operation and hence focus on combating such jamming in the remainder of this paper.

2.1.2. Techniques for Robust Transmission. The traditional defenses against jamming include spread spectrum techniques [5] where the energy of the signal is spread across a very wide bandwidth. Another important class of antijamming techniques is channel hopping [6, 7] where the signal transmission channel is changing over time. While spread spectrum and frequency hopping techniques are important physical layer mechanisms for combating jamming, additional protection is required at the packet-level. As in the case of standard wireless protocols such as IEEE 802.11 and Bluetooth, the jammer may know the pseudorandom noise code or frequency hopping sequence.

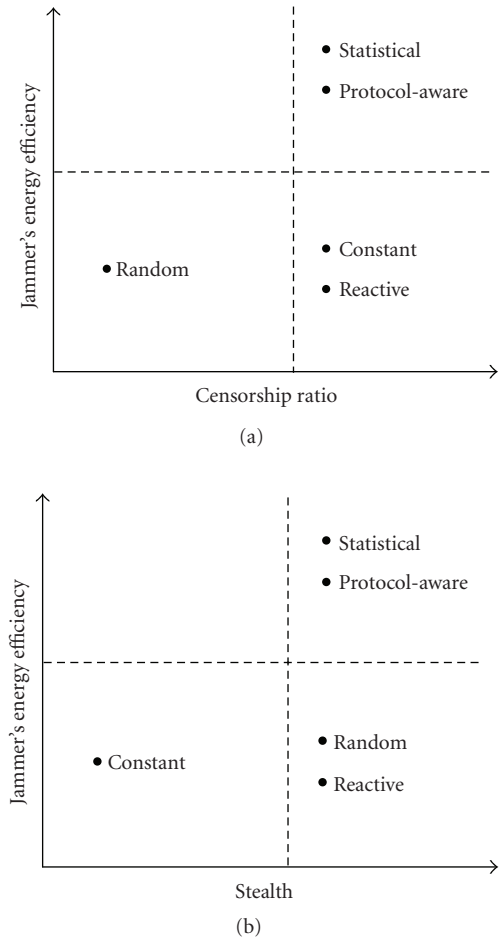


Figure 1: (a) Jammer's Energy efficiency versus Censorship ratio and (b) Energy efficiency versus Stealth.

There have been several efforts to make communication in sensor networks more robust in the presence of a jammer. In [7], Wood et al. described DEEJAM, a link layer protocol that includes several schemes for robust IEEE 802.15.4 based communication for reactive and random jammers. While mechanisms such as coding and fragmentation are proposed, the jammer still has a competitive advantage in that it may increase the power of its jamming signal, and a single jamming signal is capable of jamming multiple links in the vicinity. The authors assume that reactive jammers can be considered energy-efficient. Current radio transceivers, with the IEEE 802.15.4 physical layer of communication, use almost the same, if not greater, energy for receiving as they do for transmission [8].

In cases where resilience to jamming is not possible, it is useful to detect and estimate the extent to which the jammer has influence over the network. A jammed-area mapping protocol is described in [9] which can be used to delineate regions affected by a jammer. Such information can ultimately be used for network routing. One of the requirements of the protocol is that every node knows

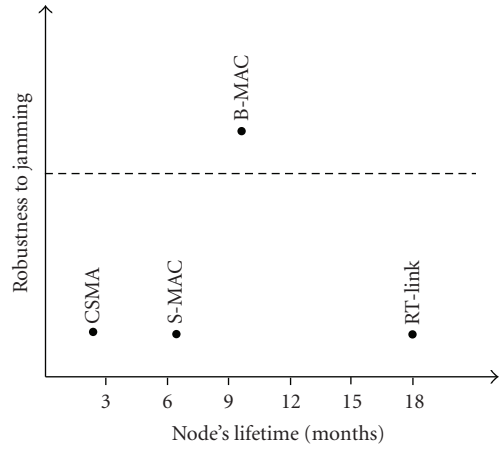


Figure 2: Comparison of robustness to jamming and energy-efficient operation of sensor MAC protocols.

its own position along with positions of all its neighbors. Our proposed solution, WisperNet, does not require such position and direction information and directly computes routes with the highest end-to-end packet delivery rate.

2.2. Impact of Jamming on MAC Protocols. We now investigate the characteristics of different classes of sensor network link protocols and the impact of a jammer on each class.

2.2.1. Energy-Efficient MAC Protocols. Several MAC protocols have been proposed for low-power operation for multihop wireless mesh networks. Such protocols may be categorized by their use of time synchronization as asynchronous [10], loosely synchronous [11, 12], and fully synchronized protocols [13, 14]. In general, with a greater degree of synchronization between nodes, packet delivery is more energy-efficient due to the minimization of idle listening when there is no communication, better collision avoidance, and elimination of overhearing of neighbor conversations.

Asynchronous protocols such as Carrier Sense Multiple Access (CSMA) are susceptible to jamming both at the transmitter (busy channel indication) and at the receiver (energy drain). The Berkeley MAC (B-MAC) [10] protocol performs the best in terms of energy conservation and simplicity in design. B-MAC supports CSMA with low-power listening (LPL) where each node periodically wakes up after a sample interval and checks the channel for activity for a short duration of 0.25 ms. If the channel is found to be active, the node stays awake to receive the payload following an extended preamble. Using this scheme, nodes may efficiently check for neighbor activity while maintaining no explicit schedule which a statistical jammer may exploit.

Loosely-synchronous protocols such as S-MAC [11] and T-MAC [15] employ local sleep-wake schedules known as *virtual clustering* between node pairs to coordinate packet exchanges while reducing idle operation. Both schemes exchange synchronizing packets to inform their neighbors of the interval until their next activity and use CSMA prior

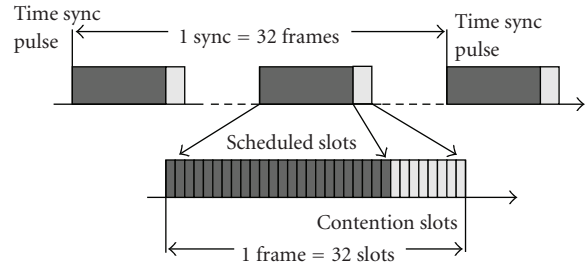


Figure 3: RT-Link time slot allocation with out-of-band synchronization pulses.

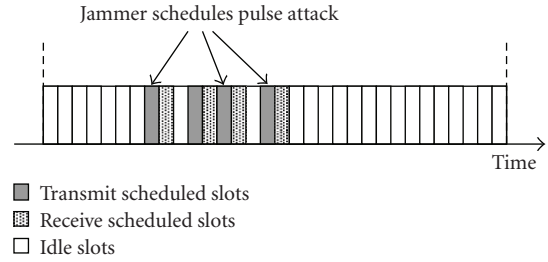


Figure 4: An attacker can learn node schedules and selectively jam active time slots.

to transmissions. S-MAC results in clustering of channel activity and is hence vulnerable to a statistical jammer.

Synchronous protocols, such as RT-Link [14], utilize hardware-based time synchronization to precisely and periodically schedule activity in well-defined TDMA slots. RT-Link utilizes an out-of-band synchronization mechanism using an AM broadcast pulse. Each node is equipped with two radios, an AM receiver for time synchronization and an 802.15.4 transceiver for data communication. A central synchronization unit periodically transmits a 50 μs AM sync pulse. Each node wakes up just before the expected pulse epoch and synchronizes the operating system upon detecting the pulse. As the out-of-band sync pulse is a high-power (30 W) signal with no encoded data, it is not easily jammed by a malicious sensor node.

In general, RT-Link outperforms B-MAC which in turn out-performs S-MAC in terms of battery life across all event intervals [14]. Figure 2 shows the relative node lifetimes for 2AA batteries and similar transmission duty cycles. Here node lifetimes for CSMA, S-MAC, B-MAC, and RT-link are 0.19, 0.54, 0.78, and 1.5 years, respectively, for a network of 10 nodes with a 10 s event sample period (based on measurement values from [11, 14]). While RT-Link nodes communicate in periodic and well-defined fixed-size time slots as shown in Figure 3, a statistical jammer is able to easily determine the channel activity schedule and duration of each scheduled transmission. As shown in Figure 4, an attacker can glean the channel activity pattern by scanning the channel and schedule a jamming signal to coincide with the packet preamble at the start of a time slot.

2.2.2. Statistical Jamming. We focus on the statistical jammer's performance with S-MAC and RT-Link as both result

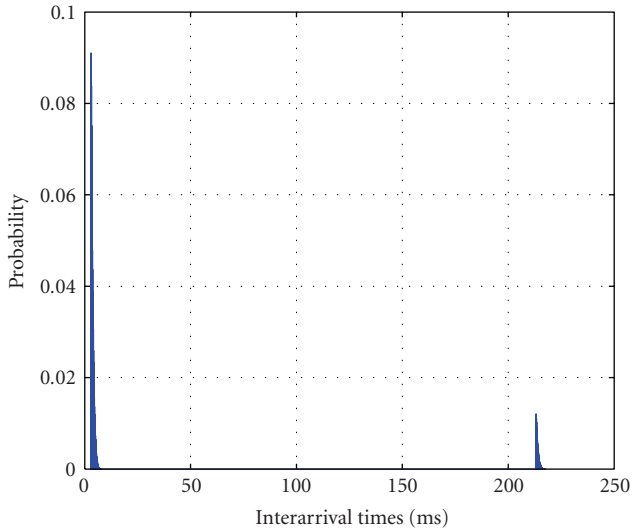


Figure 5: SMAC PDF for 15% utilization.

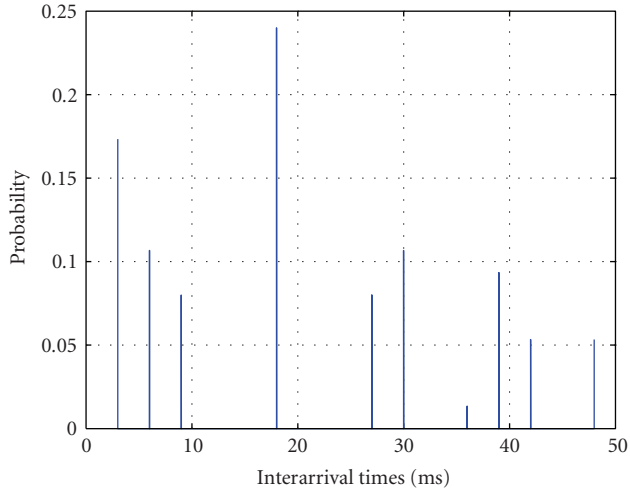


Figure 6: RT-Link PDF for 15% utilization.

in explicit patterns in packet interarrival times. We do not consider B-MAC as we aim to leverage the more energy-efficient RT-Link as a base synchronized link-layer mechanism for WisperNet. We simulated a network of 10 nodes in each case, with a 3 ms average transmission duration. In the case of S-MAC, we observe that all nodes quickly converge on one major activity period of 215 ms. In Figure 5, we also notice a spike close to 2 ms. This is the interval between the transmission of control packets and data packets at the start of an activity period. In the case of RT-Link, we simulated four flows with different rates and hence observe 4 distinct spikes in Figure 6. The other spikes with lower intensity are harmonics due to multiples of 32 slots in a frame. In both cases, we observe distinct interarrival patterns which enable a statistical jammer to efficiently attack both protocols.

In this section, we showed that statistical jammers, which exploit the observed temporal patterns in channel activity, are both more energy-efficient than reactive and random jammers and more deceptive than constant jammers. With slot schedule and slot duration randomization, the statistical jammer's efficiency is reduced to that of a random jammer. Random jammers, while relatively inefficient, are still effective with censorship that increases linearly with link utilization. To avoid a random jammer, it is essential not to schedule nodes within the physical vicinity of the jamming source. We therefore have to employ temporal randomization to combat a statistical jammer followed by spatial route adaptation to avoid the resultant random jammer. Our goal is to develop a protocol which, in the presence of a statistical jammer, has energy-efficiency near that of RT-Link and a censorship ratio lower than that of a random jammer.

Our goal with WisperNet is to develop coordinated randomized schedules, packet sizes, and routes such that a statistical jammer is unable to extract any distinctive features from the packet interarrival distribution.

2.3. Assumptions. We make several assumptions in the design and evaluation of WisperNet. We assume that the jammer is as energy-constrained as a legitimate node and must maintain a stealth operation with a low dutycycle. All packets exchanged between nodes are encrypted with a group key shared by legitimate nodes and hence the jammer is not protocol-aware. We consider both malicious and nonmalicious jamming and do not differentiate between them as the antijamming mechanisms are native to the link and network protocol. The transmission power is 0 dBm (1 mW) and in the worst case (with maximum power link jamming) the nominal packet delivery rate is never below $\widetilde{2}0\%$. This has been demonstrated in previous experiments [14]. For simplicity, we presume that the interference range is equal to the transmission range of one hop. This restriction does not limit our results. We assume that all communication is between a central gateway and each of the nodes across one or more hops.

3. Antijamming with Coordinated Spatiotemporal Randomization

An effective approach to diminish the impact of a statistical jammer on TDMA-based MAC protocols is to eliminate the possibility to extract patterns in communication. These patterns appear as a result of the use of fixed schedules which are set when a node joins a network and are assumed to repeat till the network is disbanded. Such simple and repetitive patterns are maintained with tight time synchronization and result in minimal energy consumption, deterministic end-to-end delay, and perhaps maximal transmission concurrency. In order to limit the impact of statistical jamming but still benefit from the above energy and timeliness performance, we maintain the time synchronization but change the schedule, transmission duration, and routes in a randomized yet coordinated manner along small time scales.

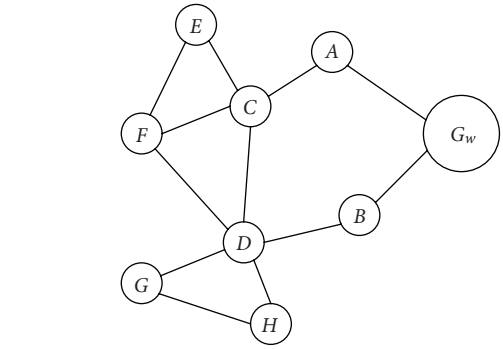
Two components of the WisperNet protocol are Coordinated Temporal Randomization (WisperNet-Time) and Coordinated Spatial Adaptation (WisperNet-Space), which perform different actions in the temporal and spatial

domains, respectively. WisperNet-Time is designed to defeat statistical jammers. By randomizing the communication in time, a statistical jammer's performance is reduced to that of a random jammer as the distribution of packet interarrival times is flat. No timing-based scheme can reduce the probability of being jammed by a random pulse jammer. In this case, the only way to decrease the jamming impact is by avoiding the jammed areas using WisperNet-Space. WisperNet-Space implements adaptive network routing as a jamming avoidance mechanism to use links which are less affected by the jammer, if possible. Both WisperNet-Time and WisperNet-Space incorporate on-line algorithms where the network is continuously monitored and node operations are adjusted in time and space.

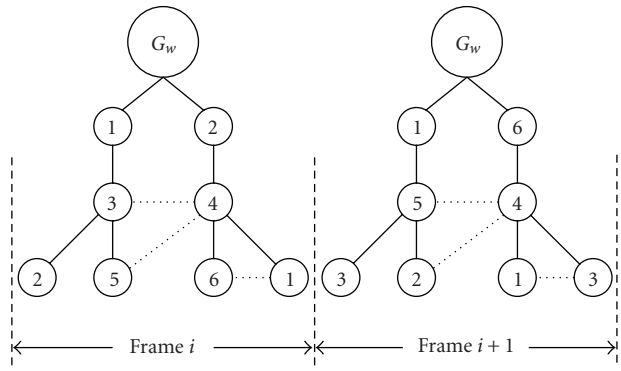
3.1. WisperNet-Time: Coordinated Temporal Randomization. The main requirement for the proposed protocol is the provision of tight time synchronization between nodes. In order to keep coordination between nodes, all nodes have to be informed about current network state in terms of current slot schedule, current slot duration, and current active network topology. We achieve this by building upon the FireFly sensor network platform [16] and using the basic synchronization mechanisms adopted in the RT-Link protocol. As illustrated in Figure 3, all communication with RT-Link is in designated time slots. 32 time slots form a frame and 32 frames form a cycle. The time sync pulse is received once every cycle. Each FireFly node is capable of both hardware-based global time synchronization and software-based in-band time synchronization. A second requirement for WisperNet is that changes in state should require minimum gateway-to-node communication and no state information exchange between nodes. All communication must be encrypted and authenticated so that an eavesdropper may not be able to extract the logical state of the network. We describe the authentication and implicit coordination scheme in the following section and the synchronization mechanism in the Implementation section.

3.1.1. Schedule Randomization. The first step toward schedule randomization is a pruning of the physical network topology graph into a directed acyclical graph. Figure 7(a) shows an example network topology graph, where each edge represents physical wireless link between two nodes. The physical network topology is logically pruned by disabling desired links. In order to logically remove a link, a node is scheduled to sleep during that particular neighbor's transmission, thereby ignoring that transmission. By forming a directed acyclic graph, we are able to efficiently assign noncolliding schedules that can be changed for every frame, as shown in Figure 7(b). Links marked by the dashed line are inactive but must be accounted for by any graph coloring algorithm.

The algorithm for schedule randomization is organized in a distributed manner. Every node uses a PseudoRandom Function (PRF) to obtain its transmission schedule from the current network key and its node ID. The transmission schedule consists of different slot indexes that can be used for transmission to neighboring nodes. The schedule changes for every frame (i.e., 32 slots) and during a frame, a node transmits only on the time-slots determined by its PRF output. After transmission, every node goes to sleep, setting its sleep timer to wake up for the earliest receive or transmit slot. In this way, energy consumption is reduced to minimum.



(a) Physical network topology



(b) Logical network topology

Figure 7: (a) Example network topology and (b) bits collision-free transmit schedule from frame-to-frame.

To obtain nonrepeating schedules, but with full coordination between nodes, the PRF computed by every node uses the current active network key along with its node ID. Once in a cycle, between two synchronization pulses, the gateway broadcasts the active keys for the next cycle. The keys, members of the one-way key chain, are generated during gateway's initialization and are stored in its memory. All keys from this chain are calculated from randomly chosen last key K_n by repeatedly applying one-way function F (as shown in Figure 8):

$$K_j = F\left(K_{j+1}\right), \quad j = 0, 1, 2, \ldots n-1. \tag{1}$$

As F is a one-way function, all previous members of chain, $(K_0, K_1, \ldots, K_{j-1})$ can be calculated from some chain element K_j but subsequent chain members $K_j, K_{j+1}, \ldots, K_n$ [17] cannot be derived. This authentication scheme is similar to [18, 19] but its use for scheduling is new.

We use the SHA1-HMAC [20] keyed-hash function to generate the current slot schedule. Therefore, for schedule computation HMAC(ID, K_j) is used, where K_j presents

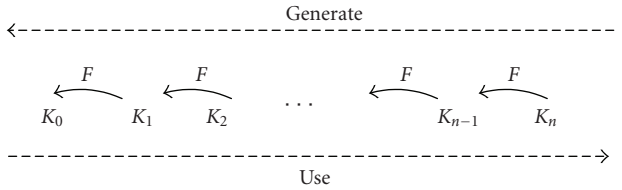


Figure 8: Generation of keys at the gateway, using a one-way hash function.

currently active network key (member of the one-way key chain). SHA1-HMAC outputs 160 bits which are used to specify the schedule of transmission slots for each of the 32 frames. These 160 bits are divided into 32 groups of 5-bits, where the node's transmit schedule in ith frame ($i = 0, 1, 2, \ldots, 31$) is determined by ith group of 5 bits. These 5 bits represent the index of one of the 32 frame's slots, eventually used for transmission.

3.1.2. Implicit Schedule Conflict Resolution. This approach for determining the transmission schedule locally can introduce a problem of potential interference that may occur when neighboring nodes are assigned the same random slot. To prevent this, every node, in addition to its schedule, calculates a slot precedence (or priority) for every transmission. The precedence for the ith frame's transmission schedule is calculated using $\mathrm{ROL}(\mathrm{reverse}(\mathrm{HMAC}(\mathrm{ID}, K_j)), 5 \cdot (i-1))$, where reverse returns 160 bits in reversed order, while $\mathrm{ROL}(w, n)$ is a left rotate of the word w for n places. Transmissions' precedences of nodes scheduled to transmit in the same slot are compared and the node with the highest precedence value is allowed to transmit. The proposed approach keeps full randomization and since $\mathrm{HMAC}(\mathrm{ID}_p, K_j) \neq \mathrm{HMAC}(\mathrm{ID}_q, K_j)$ for $\mathrm{ID}_p \neq \mathrm{ID}_q$, two nodes cannot have same transmit slot schedule with the same precedences.

To compute its schedule in one sync period with 32 frames, beside some basic bitwise operations, each node has to calculate exactly one SHA1-HMAC function for itself, and one SHA1-HMAC per node for all nodes in its k-hop interference range. We assume that the node IDs of all k-hop neighbors are known (when a node joins the network it broadcasts its ID to all its neighbors in k-hop range). Given the IDs for all nodes in its k-hop radius, a node calculates the schedule and precedence for all of neighbors as shown in Figure 9. After schedule conflicts are resolved implicitly based on the higher precedence, the node follows the combined transmit and receives schedule in a single vector. The proposed solution introduces some additional memory and processing requirements considered in detail in implementation section.

Since the full synchronization between nodes' schedules is achieved, schedule randomization does not introduce any additional energy consumption in any of the nodes, even when a node sends messages designated only to some of its neighbors. For example, consider the topology presented in Figure 7(a). If node C transmits two separate messages per frame, for both nodes E and F, there is no need for node E to be awake when a message for node F is transmitted, and vice

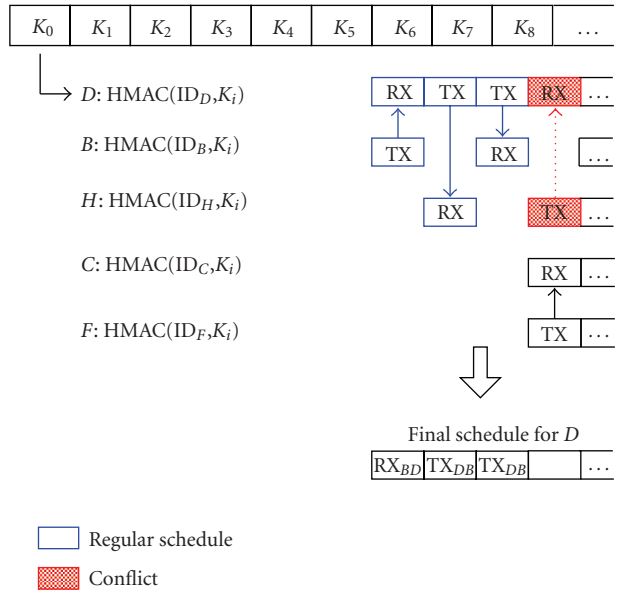


Figure 9: Implicit conflict resolution.

versa. In standard TDMA protocols, like RT-Link, these two transmissions would be dedicated to particular slots, where for example, the earlier slot is used for $C \rightarrow E$ transmission and the latter is used for $C \rightarrow F$ transmission. The same approach can be used with WisperNet. Node C calculates two SHA1-HMAC outputs, where the first one is used for $C \rightarrow E$ link scheduling, while the other is used for $C \rightarrow F$ scheduling. The above example shows that the implemented algorithm allows light-weight migration from the use of RT-Link enabled applications to use WisperNet. More details on the implementation are presented in Section 5.

The proposed slot conflict resolution can have minor inefficiencies when a node with a higher transmit precedence for a particular slot does not have any message to send, while another node is not allowed to send its message in the same slot due to a lower precedence. This results in a lower end-to-end bandwidth and an increase in a message delay. However, this issue has a fairly low probability of occurring in networks for sensor networks with a low-to-moderate duty cycle.

3.1.3. Slot Size Randomization. Even though the statistical jammer uses energy efficient pulse attacks, the proposed schedule randomization reduces its efficiency to that of a random jammer. However, with the schedule randomization, an adversary is able to estimate slot sizes from the probability distribution function (PDF) of packet interarrival times [4]. This statistical jamming scheme allows the jammer to transmit short pulse attacks at beginning of each slot, therefore corrupting all communication attempts. Although this jamming scheme is less energy efficient than a fixed schedule TDMA protocol, it is still more efficient than a random jammer.

Slot size randomization is implemented in a similar manner to slot schedule randomization, using SHA1-HMAC, as schedule randomization, but with one important difference. Instead of using the last revealed key for the slot size

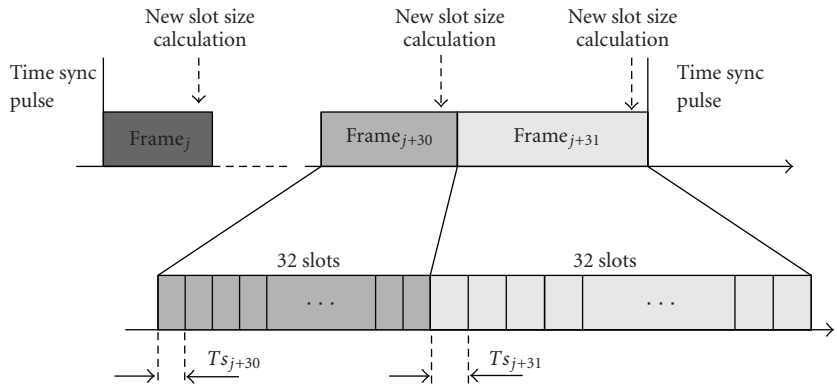


Figure 10: Slot size randomization on a frame-by-frame basis.

calculation, every node uses a shared predefined key, K_{slot}, and the network's state counter cnt. Therefore, for slot size randomization the PRF is calculated as HMAC(cnt, K_{slot}). The cycle counter is transmitted in the header of each packet and is incremented every cycle. K_{slot} is intentionally a local key so that a node joining the network will be able to synchronize its slot sizes after receiving one legitimate packet. The SHA1-HMAC's 160 bits output, calculated once per cycle, is used to calculate slot sizes on a frame-by-frame basis for all frames in a cycle, as shown in Figure 10. For example, if slot sizes can have 32 possible values, each slot size can be determined from a group of 5 bits. Thus, slot sizes for all 32 frames between two sync pulses can be calculated from SHA1-HMAC's 160 bits output.

The network's state counter represents the number of sync pulses received by the network, and its value is exchanged between neighboring nodes in the header of every packet. Here we assume that every sync pulse is received as it is a global and high-power AM pulse. Therefore, it is only nodes who want to join an already operational network need to be informed about current network counter. The proposed coordinated slot size randomization scheme assures that all nodes know the current frame's slot sizes and allows them to calculate an accurate time interval for their transmissions/receptions.

If a key from the key chain is used for slot size calculation instead a predefined one, in cases when a node does not receive a key from the gateway would result in complete loss of synchronization. Without correct information about a slot size, nodes that do not know the current frame size are not able to schedule themselves to wake-up for the expected sync pulse. With the predefined key used for slot size calculations, nodes are always able to know size of each frame, therefore, they can schedule their awakening on time.

Slot sizes have values from a discrete set, where the set size is determined by the number of PRF output bits. The number of values used for slot sizes and relative distance between them have direct influence on PDF of packet interarrivals times. The goal of our antijamming scheme is to have a uniform PDF, or at least a PDF with spikes flattened as much as possible, which does not allow timing information extraction. It is recommended that at least 8 slot sizes with small relative difference between them be used.

Slot size randomization requires additional memory resources if nodes need to send some fixed-size data block in a fixed time interval. In this case slot sizes can be both smaller and bigger than the size necessary for data block transmission, which can result in lower network utilization in former case or data congestion in later case. In the latter case, a portion of the data available for transmission will have to be buffered in node's internal queue till the next transmission slot occurs. In this case, the average slot size must be larger than slot size needed for one data block transmission. The size of the queue needed in every node is directly connected with ratio between these two sizes.

3.2. WisperNet-Time: Performance Analysis. We now investigate the impact of channel utilization on the PDF of packet interarrival times. We also determine the buffering needs due to randomized slot sizing and its impact on the end-to-end delay. Finally we look at the censorship ratio versus jammer's lifetime for RT-Link, S-MAC, and WisperNet.

We conducted a simulation in Matlab on a protocol with structure similar to RT-Link [14], where each cycle consists of 32 frames and each frame consists of 32 slots. At the beginning of every cycle, a sync pulse is transmitted. We have simulated a system where slot sizes have uniform distribution with values in range [15] ms. A maximum slot size of 5 ms is chosen to match the maximum message size of 128 bytes for IEEE 802.15.4 transceiver with data rate of 250 kbps[8]. 128 bytes can be sent with transmission duration of 4.2 ms and the rest of the slot time is used for interslot processing and for guard times. A simulation for 10000 sync pulses (i.e., cycles) was carried out, which on an average lasts 50 minutes.

We first simulated the influence of the link utilization factor (U) on the PDF of interarrival times and show that it has very little influence with the proposed scheme. This is one of the major benefits of WisperNet-Time, because for other protocols the only way to reduce spikes in the PDF is to reduce the utilization factor, as proposed in [4]. Results for PDF of interarrival times are shown in Figure 11 for U = 50%, where slot sizes were randomly chosen from one of the 32 possible values in desired span. While channel utilization has an effect to the PDF, it is to a significantly smaller extent than in B-MAC's case. We observe that the peak of the PDF is less than 2% and no patterns can be extracted by the jammer.

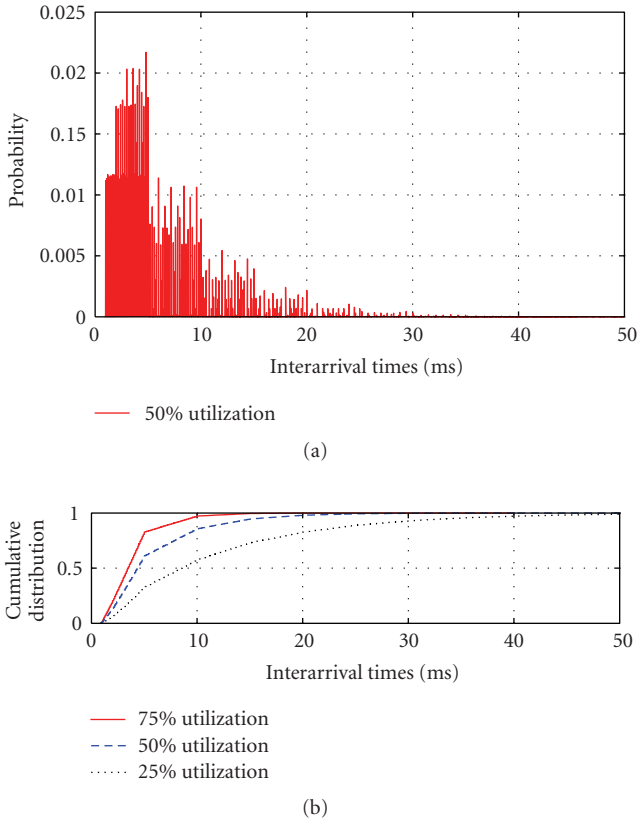


Figure 11: (a) PDF of interarrival times for WisperNet. (b) Corresponding CDF.

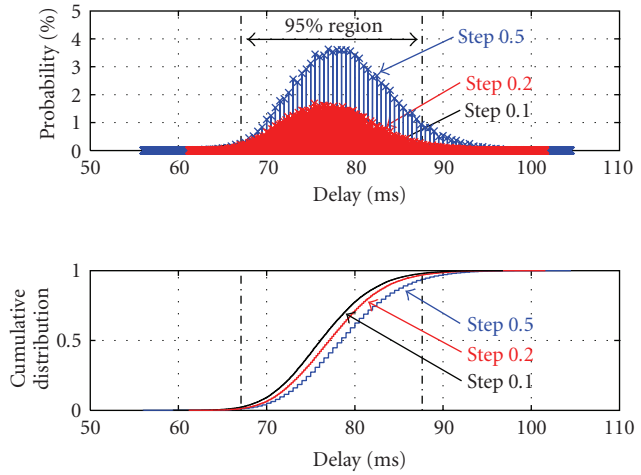


Figure 12: Message delay and its Cumulative Distribution at last node, for 20 nodes chain.

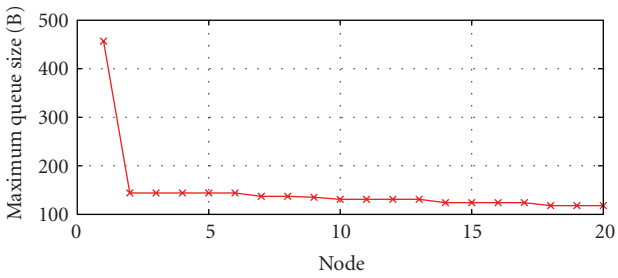


Figure 13: Maximum queue size for each node in 20 nodes chain.

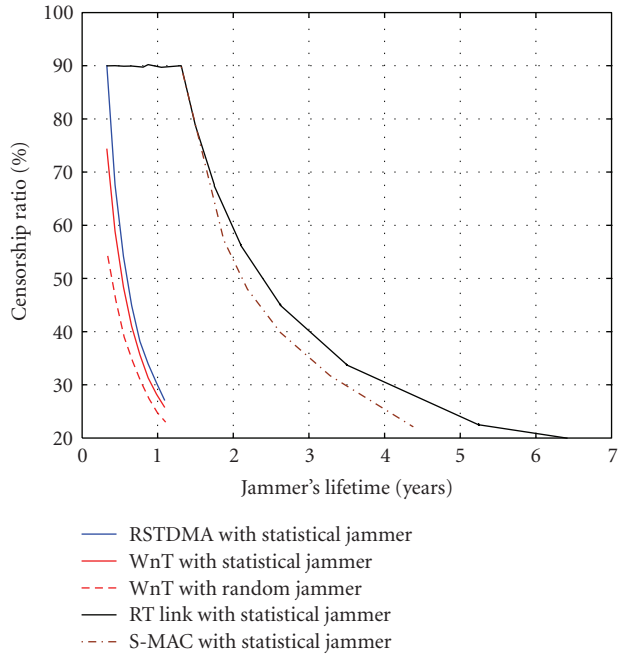
With U below 50%, a small increment can be seen on spikes in (5 10] ms interval. Also with integer multiples of some slot sizes within [15] ms interval, influence of U can almost be ignored. Due to the uniform distribution for all three cases of U, it is not possible to extract slot sizes. Even if the PDF is derived from a smaller statistical sample size, the results are similar due to the pseudorandomness of the slot size.

3.2.1. Impact on End-to-End Delay. To analyze how the slot size changes affect the end-to-end delay, we simulated a 1-dimension chain with 20 nodes. In every frame, each node forwards the previously received data to next node. If the slot size is smaller than necessary to transmit the backlogged data, the maximum allowed packet size is sent and a rest of the data is buffered. In a simulated chain, the source node receives fixed size data blocks at a fixed interval and at a slightly smaller size than for an average slot size (e.g., 3 ms).

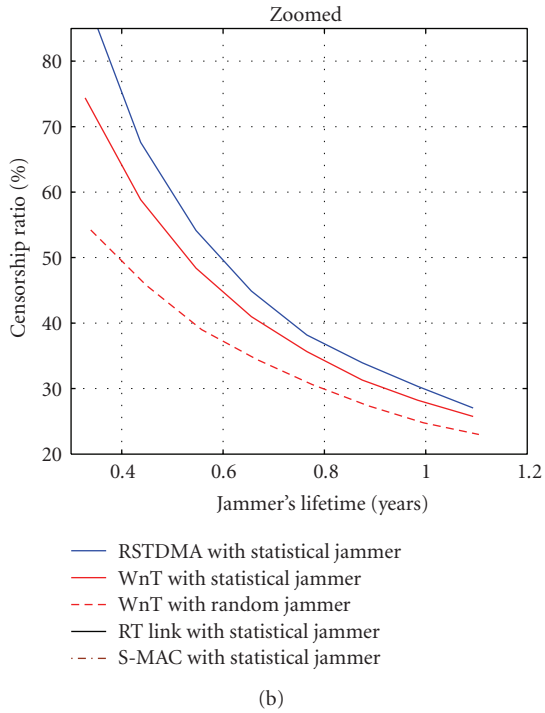
Figure 12 presents PDF and CDF of delay at last node for different slot's size quantizations. We observe that with finer quantization of slot sizes, the distribution interval of the last node has not only a smaller maximum delay but also a smaller possible set of delay values that can be expected. The reason is that finer quantization allows better data distribution per packets and therefore reduces the delay caused with packets fragmentation. Expectedly, this randomization introduced some additional delay. In a TDMA system with a constant slot size of 3 ms, the delay at the last node would be $20 \cdot 3\,\text{ms} = 60\,\text{ms}$. Here average delay is around 75 ms, but with 95% probability interval [67–88] ms. This shows that randomization introduces some variability into the communication end-to-end delay estimation.

3.2.2. Impact on Memory Requirements. Another potential side-effect of WisperNet-Time is a need for additional memory for data queuing. In Figure 13, the maximum queue size for every node in chain is presented. The first node, with its constant data block input, requires a buffer with an additional 512 bytes of memory. All other nodes require an additional buffer for one maximum sized packet.

3.2.3. Comparative Analysis of MAC Protocols. Figure 14 presents the relationship between the jammer's lifetime (LIFE) and censorship ratio (CR) for communication in its range. We simulated the influence of two types of jammers—statistical jammers (SJ) and random jammers (RJ)—on different kinds of protocols. Both these types of jammers transmit 150 μs-long pulse attacks. We modeled these attacks with a 90% success rate of packet corruption in cases when jamming pulse is transmitted during a node's communication. For RT-Link and WisperNet-Time, we used the previously described protocols; while for the Random Schedule TDMA (RSTDMA), we also used 32 slots per



(a)



(b)

Figure 14: (a) Dependency between jammer's lifetime and Censorship Ratio and (b) zoomed.

frame, where every slot is 3 ms long. All protocols are simulated for systems with 25% of network utilization.

As we expected, for RT-Link and S-MAC, the SJ's lifetime is very high. As shown, RSTDMA is easily jammed and the SJ enjoys the longest lifetime. In addition, the slot size randomization component decreases the jammer's lifetime, for almost 0.1 years at 50% CR. Note that differences between LIFE-CR curve for SJ and RJ are caused only by the fact that SJ does not transmit pulses in intervals smaller than 1 ms, which, in this case, is the smallest slot size (only parameter that can be extracted from input signal statistics). We observe that schedule randomization has a significantly higher impact on the jammer's lifetime than does slot size randomization. This justifies our decision to use a prestored key for the latter's calculations, while using keys from the gateway's one-way chain for former. If some nodes are captured and compromised, only the predefined slot-size key would risk being extracted.

4. WisperNet-Space: Coordinated Spatial Adaptation

We now discuss spatial aspect of antijamming. For the WisperNet-Space, we consider a dense sensor network where each node is modeled as a unit disk graph. The network is represented as an undirected graph $G(V, E)$, where V is a set of nodes (vertices) and E a set of links (edges). For each link $e = e(u, v)$, k weights (or costs) $w_j(u, v)$, $(j = 1, 2, \ldots, k)$ are associated. For a tree T in graph G, the aggregate weight $W_j(T)$ is defined as

$$W_j(T) = \sum_{e \in T} w_j(e), \quad j = 1, 2, \ldots, k. \tag{2}$$

Weights associated with each link describe the different types of costs which may include the network's parafunctional properties, such as reliability of network communication or delay and energy consumption of a sensor network.

In general, a set $L \subseteq V$ of terminal nodes is given and the objective is to find a connected subgraph, spanning all the terminals with minimal aggregate weights for all $j = 1, 2, \ldots, k$. If only one weighting function is considered, $L = V$, and the connected subgraph is required to be a tree, then the problem is defined as a Minimum Spanning Tree problem (MST). The MST problem can be solved using known algorithms (Kruskal's, Boruvka's, etc.) [21]. If $L \neq V$ and also only one weighting function is considered, the problem is equivalent to Steiner minimal tree problem (SMT). SMT is an NP-complete problem, but several heuristics exist which resolve SMT problem in both a centralized and a distributed manners. For applications where more than one weighting function is to be defined, algorithms for multiconstrained routing are used in order to control all application's important parafunctional properties of the network.

For WisperNet-Space, we only considered network's reliability, so we associate a reliability weight function for each link in network. We define the weight of each link to be a function of the packet loss ratio and hence aim to derive routes which connect all essential nodes using most reliable links. The continuous execution of the cost minimization function essentially allows evasion of links under the influence of a random jammer.

The network's reliability is measured in terms of its Packet Delivery Ratio (PDR) which is defined as the ratio of packets that are successfully delivered to a destination

[3]. PDR can be perceived as the probability of error-free communication between two nodes. Thus, the reliability of a path P in the given network can be defined as a $\prod_{e\in P}\mathrm{PDR}(e)$. Our goal is to achieve maximum reliability on a path P, which is equivalent to maximizing

$$\ln \prod_{e\in P}\mathrm{PDR}(e) = \sum_{e\in P} \ln \mathrm{PDR}(e). \tag{3}$$

With $\mathrm{PDR}(e) \leq 1$ for path P, our goal is to minimize $\sum_{e\in P} |\ln \mathrm{PDR}(e)|$. Therefore, the reliability weight for some link $e = e(u, v)$ is defined as

$$w_r(u, v) = |\ln \mathrm{PDR}(u, v)|. \tag{4}$$

4.1. Active Topology Update. For adaptive routing in WisperNet-Space, we use an MST-Steiner heuristic to solve the SMT problem. All active nodes periodically send the PDRs for all their active links to the gateway. After receiving a link's weight, the gateway updates its weight table for all existing links in the network. Since the PDR is not defined for inactive (not used) links, these links keep same weights as they had prior to activation of the present network topology. Their weights cannot be reset to zero, since that would allow some heavily jammed links to become competitive for network routing right in next iteration.

In order to defend against mobile jammers, the weights of unused network's links are processed in time with a leaky integrator. To avoid situations where some previously heavily jammed link still has a high weight although the jammer that caused it has moved away, for every link $e = e(u, v)$ and the current active subgraph T, the reliability weight for the next network topology calculation is defined as

$$w_r(u, v) = \begin{cases} |\ln \mathrm{PDR}(u, v)|, & e \in T, \\ \rho \cdot w_r(u, v), & e \notin T. \end{cases} \tag{5}$$

ρ $(0 < \rho < 1)$ is a leaky constant that determines a speed of network's adaptation to jammers' mobility. It is not recommended to set a too small value for ρ, since something similar to previously described situation can happen, when a jammed link can be repeatedly included in active topology after very short duration. For example, with $\rho = 0.8$ reliability weight for unused link would be reduced by 20% for every calculation of network topology, which would allow inclusion of the jammed link into new topology after only a few iterations. If all jammers have fixed positions, ρ can be set to 1.

After updating its weights table, the gateway calculates the new active topology with minimum costs to reach all Steiner points (i.e., active nodes). To distribute the information about active links, we used the Prufer code (sequence) [21], a unique sequence associated with a tree, which for a tree with N vertices contains $N - 2$ elements. In addition to this code, we send a second code sequence that maps the node ID of the active nodes to the index of the $N-2$ sequence. In a case of dense networks, with N nodes, from which the Steiner tree is derived, a much smaller number of nodes (M) may be active. Therefore, for all M nodes from the Steiner tree, different temporal IDs are assigned from $1 : M$ interval, and for that tree, a Prufer code with $M - 2$ elements is derived. Along with this sequence and number of active nodes, M, a lookup table with size M is sent, where ith position in this table contains ID of a node, that is indexed as i while creating the Prufer code sequence. In this way only $2 \cdot M - 1$ values are sent from gateway and can be encapsulated within one maximum-sized 128 byte IEEE 802.15.4 packet.

```
while 1 do
  if NewWeightArrive then
    Update Weight Table
    if AllReceived or TopologyTimerOn then
      TableUpdated ← 1
    end if
  end if
  if TableUpdated then
    SMT
    CalculateSpreadingSchedule
    FloodNetwork
  end if
  end while
```

Algorithm 1: Gateway procedure description.

4.2. Topology Maintenance and Updates. WisperNet-Space computes a new network topology every 128 cycles. Given the average slot size of 3 ms and 1024 slots/cycle, the topology update occurs every 6.4 minutes on average. The current active topology includes a subset of the node population as *active nodes* and the unused node, which are not part of the active topology, and are considered *inactive nodes*. The key challenged during a topology update is to activate the inactive nodes, which to save energy operate at a very low duty cycle.

At the beginning of the new topology distribution, all currently active nodes are informed about new active topology by a broadcast from the gateway. To activate inactive nodes, which are to be part of the topology update, 8 slots after the sync pulse are reserved for asynchronous communication. We refer to these 8 slots in each cycle to be the "topology configuration" frame. Thus topology maintenance and updates account for a 0.78% overhead. Instead of using contention based transmission for inactive nodes, we opted to schedule these transmissions from inactive nodes for a more deterministic behavior for the propagation of topology updates. As information needs to be spread in only one direction (from gateway) through a low-degree tree, and since only nodes that need to activate some inactive nodes (on the periphery of the current topology) would be scheduled for transmission in these slots, the TDMA configuration frame of 8 slots is enough for collision-free transmission scheduling of a degree-4 tree with 2-hop coloring. These transmission indices are calculated using conservative version of MAX [22] for maximal transmission concurrency.

Algorithm 1 describes the gateway's procedure for topology dissemination. The generated message with the information about new network topology is distributed over the

```
while 1 do
  if mod(cnt, 2^10) == 0 and Need2Wake then
    update ← 1
  end if
  if update and MyTxConfigSlot then
    Send Message
  end if
  if update and AllChildrenRetransmitted then
    update ← 0
  end if
  end while
```

Algorithm 2: Active node's procedure for new nodes activation.

```
SetWakeUp on ConfigurationSlots
 while 1 do
    NOP
 end while

 WakeUpRoutine
 if WakeUp then
    Listen
    if ReceivedMyID then
       Active ← 1
    end if
 end if
 end of WakeUpRoutine
```

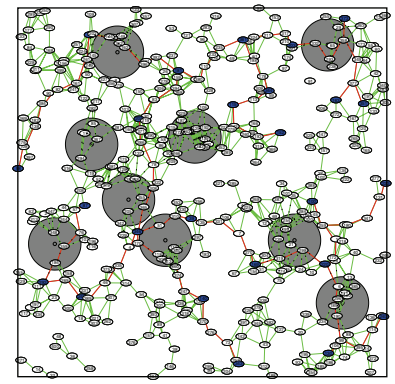
Algorithm 3: Inactive node procedure for its activation.

network using all active links. Since some currently inactive nodes may be part of next active topology, the mechanism to inform them is described in Algorithm 2 (cnt is the value of the sync pulse counter).

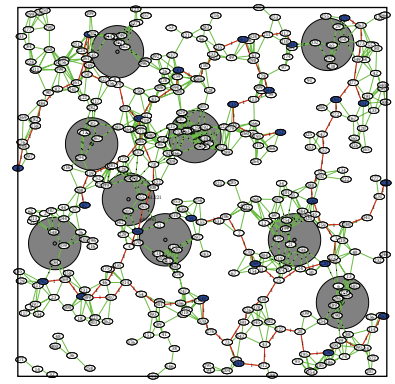
All nodes that are used for activation of the inactive nodes along with the new topology receive 3-bit index of the slot dedicated for its transmission, in the 8 slots "configuration" frame. Using this index, each node schedules its transmission of the new configuration to neighboring node and keeps on transmitting it on the same slot in every "configuration" frame. Once all its required neighboring nodes (i.e., currently active or inactive nodes that need to be activated) are heard retransmitting the new topology update, a node is assured that its topology has been successfully updated.

All inactive nodes wake up after every sync pulse, and listen for the first 8 slots in a cycle. If the message for its activation is received, a node switches to active mode and executes the active mode's algorithm (see Algorithm 3).

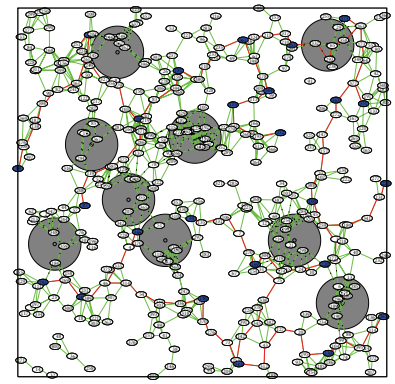
Since a new topology is computed once in 128 cycles, 1024 configuration slots are available for both activation of dormant nodes and also for association of newly added nodes. Our experiments showed that for networks with less than 500 nodes all inactive nodes are activated in first 10% of these slots. Given this, we allowed last 20% of these slots (i.e., configuration slots 820–1024) to be used as contention slots for admission of new nodes only. These slots enable nodes that want to join a network to announce their presence



(a) Network topology - beginning



(b) Network topology at an intermediate time instance t_1



(c) Network topology - optimal

Figure 15: Network topology; green links—actual, "physical" links; red—links used for Steiner tree; blue nodes—terminal nodes (members of set L); dark gray area—area under jammers' influence.

to neighboring nodes (via a HELLO packet in RT-Link), so that they can be initially included as inactive nodes in the network. RT-Link supports both contention and contention-free slots and thus makes it convenient to include this asynchronous traffic.

4.3. WisperNet-Space: Performance Analysis. In order to evaluate the performance of WisperNet-Space under random jamming attacks, we first simulated an SMT network with a random topology. A network with 400 randomly

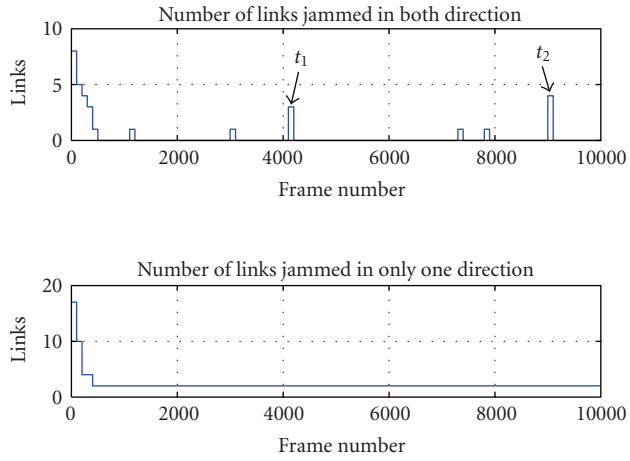


Figure 16: Number of links jammed in 2 (1) direction used for communication.

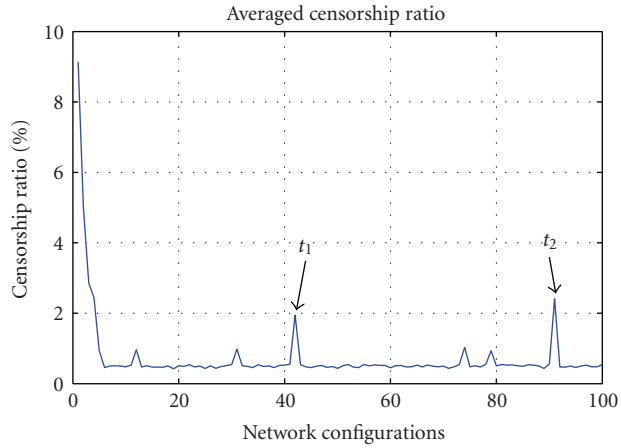


Figure 17: Averaged network's censorship ratio on 100 blocks, for implemented Steiner tree.

distributed nodes in a 4 km × 4 km square was analyzed. We also randomly distributed nine jamming nodes, each with the same RF characteristics as network's nodes. To emphasize the jamming effect in order to test WisperNet-Space's adaptation, the jammers' link utilization is set to 50%. We implemented a communication protocol so that all neighboring nodes exchange exactly one message per frame. Changes in network routes for both SMT and MST components are performed once in 100 frames. In our experiments, we used a value of 0.999 for ρ, which decreases reliability weight of unused link by 1% for every period of 96 seconds (on average).

Figure 15(a) presents the initial network topology and the initial routes. The terminal nodes and the areas under attack by the jammers are highlighted. We observe that a large number of active links are under attack. The average censorship ratio for this network is 9% for this startup configuration. The censorship ratio decreases to less than 1% as the routes adapt to more reliable paths and it can not go below this minimum value. This is because for the given Steiner tree topology and its distribution of jamming regions, the best case routes determined by WisperNet-Space do include at least 2 partially jammed links. The optimal configuration includes 0 links jammed in both directions and 2 links jammed in only one direction, as shown in Figure 16.

We observed that at some intermediate moments (e.g., moments t_1 and t_2 as seen in Figure 15(b) and corresponding Figure 17), network routes with more jammed links were chosen, which directly resulted in an increase of the overall censorship ratio. This is more prominent at the beginning of the WisperNet-Space's operation when all links start with the same minimum weight (0). We see in Figure 15(b), that three links are jammed in both directions in the top-right corner. As the routing algorithm explores the problem space with different sets of active links, it often chooses links under heavy influence of the jammer. As this procedure of refining the route continues and more links are evaluated for the first time, the algorithm will choose jammed links only if its weight, due to the leaky integrator, drops below a threshold that would make the aggregate weight of a subgraph smaller than the aggregate weight of currently used subgraph. We observe these spikes in the network's censorship ratio in Figure 17. Figure 15(c) presents optimal solution where only two links from highlighted area, jammed in only one direction, are used for routing. Over the course of the adaptation for one hour, we observe in Figure 18 that the number of active links does not vary much. We also noticed that the stretch factor of the network path lengths is ≤1.3 due to the end-to-end weight minimization function for calculating cumulative packet reliability across multiple links.

5. Implementation and Evaluation

The WisperNet antijamming protocol was implemented on a network of FireFly sensor nodes [16]. Each node consists of a microcontroller, an 802.15.4 2.4 GHz transceiver, and multiple sensors. Figure 19 shows two configurations of the FireFly node—one with in-band software-based time synchronization and the other with an add-on AM radio receiver for receiving an out-of-band AM sync pulse. In order to achieve the highly accurate time synchronization required for TDMA at a packet level granularity, we use a carrier-current AM transmitter to provide an out-of-band time synchronization pulse. The time synchronization transmitter, a separate module, is plugged into the wall-outlet and uses the building's power grid as an extended AM antenna. This scheme was able to cover an entire eight story building. Nodes were synchronized with a 50 μs pulse that was transmitted every 10 seconds from the AM transmitter (see [16] for details). The AM pulse has a jitter of ≤150 μs.

We implemented our jamming avoidance scheme incorporating SHA1, SHA1-HMAC, gateway schedule updates, and neighbor information exchange in 8-bit fixed-point C for the Atmel ATMEGA32L microcontroller and a 16-bit 16 MHz TI MSP430F22x microcontroller. Each node ran the nano-RK [23] real-time operating system and the RT-Link [14] link protocol. The RT-Link TDMA cycle includes 32 frames which in turn are composed of 32 slots. The slot sizes were assigned values from [15] ms. In our tests, every node attempts to transmit one message per frame.

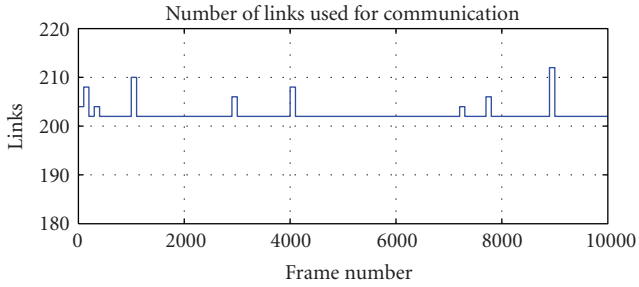


Figure 18: Number of active links in the network for implemented Steiner tree.

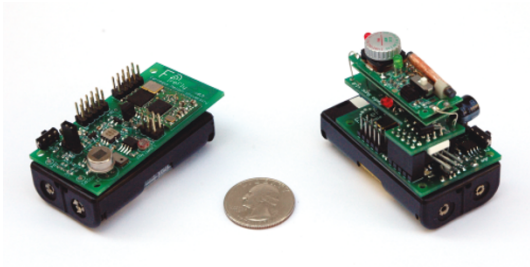


Figure 19: FireFly node with sensors on the left; on the right FireFly node with add-on radio receiver.

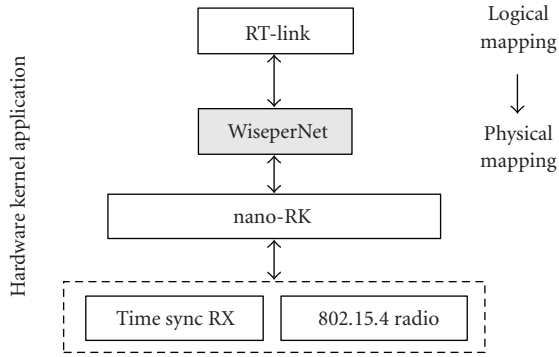


Figure 20: WisperNet application layer.

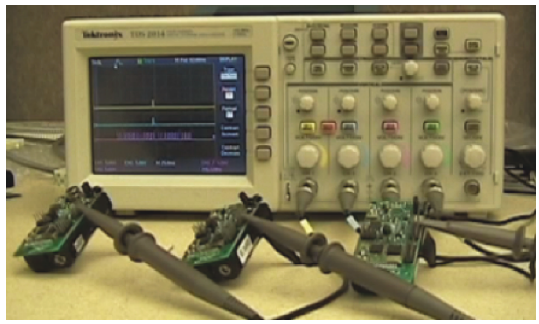


Figure 21: Experimental setup with 3 FireFly nodes.

We observe that most of the current hardware platforms used for wireless sensor networks development are not CPU-constrained, but have a memory limitation. Our implementation of the SHA1-HMAC function required only 3 additional 160-bit buffers as the comparison of schedules and precedences is done iteratively for all the node's neighbors. The SHA1-HMAC function required 12.5 ms for calculations on TI's MSP430F22x microcontroller. For networks where maximal node's degree in a network is N, every node, in the worst case, needs to execute SHA1-HMAC function $(N^2 + 1)$ times for schedule calculation and once more for slot size computation in every sync period. For example, if $N = 5$, in worst case, 27 SHA1-HMAC function executions are needed, which results in 337.5 ms of CPU time used for these calculations in every sync period, where one sync period contains $32 \cdot 32 = 1024$ slots, with sizes from 1 ms to 5 ms. Therefore, in the worst case, less than 33% of CPU processing is used for these calculations, while on average less than 11% is used. From perspective of memory constraints, the implemented procedure for schedule randomization uses 276 bytes of flash memory and 400 bytes of RAM. Since only two nodes' schedules have to be compared in each iteration, the schedule calculation procedure occupies only 236 bytes for code size. For the initialization vector of the SHA1-HMAC function, 40 bytes of data are prestored in FLASH. As same code is used for slot size randomization, and since this procedure is called after previous calculations, no additional memory is needed for slot size computation.

WisperNet is implemented in the application layer, on top of the nano-RK kernel [23], as shown in Figure 20. To allow an easy transition for applications that exploit RT-Link, WisperNet can be considered as a "slot remapping layer" below RT-Link. With this, the API for RT-Link based applications is used only for logical slot ordering, while the actual logical schedule mapping to the current set of randomized slots is performed by WisperNet. As in the example presented in Figure 7, WisperNet just uses ordering relations between RT-Link transmission slots' indexes. This keeps connectivity between all pairs of nodes unchanged and has limited impact on end-to-end delay of the RT-Link protocol.

In Figure 21, we connected three nodes to the oscilloscope to display the transmit and receive activity. Two nodes were programmed to be a transmit and receive pair to show the coordinated and collision-free schedule randomization. A third node was programmed to raise a signal on every slot to provide a reference of the slot boundaries. In Figure 22, the top signal on the oscilloscope is triggered by the transmit pin of the transmitter and the middle signal is triggered by the receive pin on the receiver. The signal at the bottom is triggered on every slot interval to provide a reference of the slot boundaries. We observe that the schedule is both coordinated and changes on a frame-by-frame basis.

Finally, we evaluate the impact of clock drift on the proposed scheme. Synchronization pulses are transmitted every 10 seconds to maintain synchronization between nodes. In [16], the authors have observed that the worst case clock drift of FireFly's clocks was 10 μs/s, equivalent to clock accuracy of 10ppm (parts per million). The drift of the clock crystals has been observed to be relatively regular. Since the AM transmitter's pulse jitter is less than 150 μs, the time between synchronization pulses is measured and used for software clock-rate adjustments on the FireFly

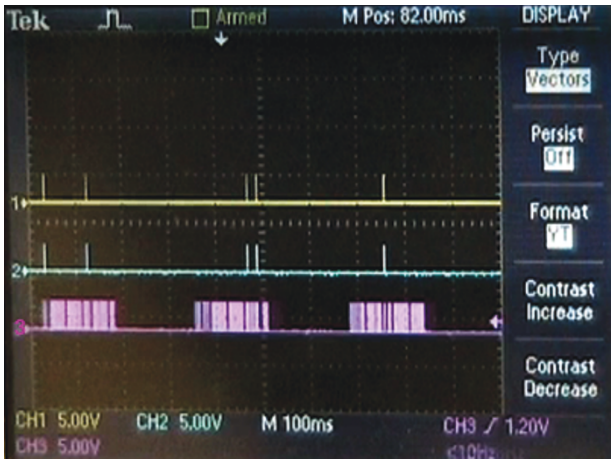


Figure 22: Implementation of WisperNet-Time showing changing patterns on a frame-by-frame basis.

node. This clock-rate adjustment procedure achieves a global synchronization accuracy within 120 μs. Therefore, by setting the guard time for each time slot to be 0.5 ms, 150 μs AM pulses' drift allows a node to miss up to three consecutive AM sync pulses without loosing the synchronization. Since 99.6% of the synchronization pulses are properly detected [16], the probability that more than three pulses will not be received is less than $3 \cdot 10^{-10}$. However, even if this occurs, the node is able to resync again, after it receives a sync pulse.

5.1. Limitations. We observe some limitations with our current implementation. The WisperNet-Spatial routing scheme is centralized and will not scale well in large networks (>500 nodes) under moderate to heavy attack as the message from the gateway may not get through. While several distributed heuristics for the MST problem exist, they require a large amount of information with respect to shortest paths from a node to all other nodes in the network. For continuously changing system, with respect to the weights of used link, these distributed solutions do not represent satisfactory solution. Also these schemes are not conducive to energy-constrained and memory-constrained sensor networks. We aim to explore distributed solutions further. The presented solution is intended to serve as a reference point, a step toward distributed solution for spatiotemporal antijamming.

A second limitation is that all nonactive nodes in the MST are required to receive for few contention slots (i.e., 8 in our implementation) every cycle to receive and forward a route update message. This may be wasteful if the network is not under heavy attack from jammers.

6. Discussion and Conclusion

In this paper, we proposed the WisperNet antijamming protocol which is a distributed technique for Coordinated Temporal Randomization of transmissions (WisperNet-Time) to reduce the censorship ratio of a statistical jammer to that of a random jammer. A second component of WisperNet is Coordinated Spatial Adaptation (WisperNet-Space), where network routes are adapted continually to avoid jammed regions (and hence random jammers) and select paths with the max packet delivery ratio.

Through simulation and experiments, we demonstrate that WisperNet is able to effectively reduce the impact of statistical and random jammers. Unlike coding-based schemes, WisperNet is resilient to jamming even under moderate to high link utilization with ≤2% censorship rate for the network topologies explored in this work. The schedules derived from WisperNet are nonrepeating, with randomized packet lengths while maintaining coordinated and collision-free communication. WisperNet has been implemented on a network of FireFly sensor nodes with tightly synchronized operation and low operation overhead.

Acknowledgments

This work is partially supported by the NSF CPS Grant no. 0931239 and MRI Grant no. 0923518.

References

[1] M. Pajic and R. Mangharam, "Anti-jamming for embedded wireless networks," in *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN '09)*, pp. 301–312, 2009.

[2] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '05)*, pp. 46–57, 2005.

[3] W. Xu, T. Wood, W. Trappe, and Y. Zhang, "Channel surfing and spatial retreats: defenses against wireless denial of service," in *Proceedings of the 3rd ACM Workshop on Wireless Security (WiSe '04)*, pp. 80–89, 2004.

[4] Y.-W. Law, L. Van Hoesel, J. Doumen, P. Hartel, and P. Havinga, "Energy-efficient link-layer jamming attacks against wireless sensor network mac protocols," in *Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '05)*, pp. 76–88, 2005.

[5] A. J. Viterbi, "Spread spectrum communications: myths and realities," *IEEE Communications Magazine*, vol. 40, no. 5, pp. 34–41, 2002.

[6] V. Navda, A. Bohra, S. Ganguly, and D. Rubenstein, "Using channel hopping to increase 802.11 resilience to jamming attacks," in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM '07)*, pp. 2526–2530, 2007.

[7] A. D. Wood, J. A. Stankovic, and G. Zhou, "Deejam: defeating energy-efficient jamming," in *Proceedings of the 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '07)*, San Diego, Calif, USA, June 2007.

[8] Texas Instruments Inc., Chipcon CC2420 Data Sheet, 2003.

[9] A. D. Wood, J. A. Stankovic, and S. H. Son, "JAM: a jammed-area mapping service for sensor networks," in *Proceedings of the 24th IEEE Real-Time Systems Symposium (RTSS '03)*, pp. 286–297, 2003.

[10] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 95–107, 2004.

[11] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '02)*, vol. 3, pp. 1567–1576, 2002.

[12] A. El-Hoiydi and J.-D. Decotignie, "Wisemac: an ultra low power mac protocol for the downlink of infrastructure wireless sensor networks," in *Proceedings of the 9th International Symposium on Computers and Communications (ISCC '04)*, vol. 1, pp. 244–251, 2004.

[13] L. F. W. van Hoesel and P. J. M. Havinga, "A lightweight medium access protocol (lmac) for wireless sensor networks," in *Proceedings of the 1st International Workshop on Networked Sensing Systems (INSS '04)*, 2004.

[14] A. Rowe, R. Mangharam, and R. Rajkumar, "RT-Link: a time-synchronized link protocol for energy-constrained multi-hop wireless networks," in *Proceedings of the 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks (SECON '06)*, vol. 2, pp. 402–411, 2006.

[15] T. van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*, pp. 171–180, 2003.

[16] R. Mangharam, A. Rowe, and R. Rajkumar, "FireFly: a cross-layer platform for real-time embedded wireless networks," *Real-Time Systems*, vol. 37, no. 3, pp. 183–231, 2007.

[17] B. Schneier, *Applied Cryptography*, John Willey & Sons, New York, NY, USA, 1996.

[18] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, 2002.

[19] A. Perrig, R. Canetti, D. Tygar, and D. Song, "The TESLA broadcast authentication protocol," *Cryptobytes*, vol. 5, no. 2, 2002.

[20] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: keyed-hashing for message authentication," *RFC*, vol. 2104, pp. 1–12, 1997.

[21] B. Y. Wu and K. Chao, *Spanning Trees and Optimization Problems*, Chapman and Hall, CRC Press, Boca Raton, Fla, USA, 2004.

[22] R. Mangharam and R. Rajkumar, "MAX: a maximal transmission concurrency mac for wireless networks with regular structure," in *Proceeding of the 3rd International Conference on Broadband Communications, Networks and Systems (BROADNETS '06)*, 2006.

[23] A. Eswaran, A. Rowe, and R. Rajkumar, "Nano-RK: energy aware resource centric RTOS," in *Proceedings of the 26th IEEE International Real-Time Systems Symposium (RTSS '05)*, Miami, Fla, USA, December 2005.