4-2009

Anti-Jamming for Embedded Wireless Networks

Miroslav Pajic
University of Pennsylvania, pajic@seas.upenn.edu

Anti-Jamming for Embedded Wireless Networks

Abstract

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. In the context of energy-constrained wireless sensor networks, nodes are scheduled to maximize the common sleep duration and coordinate communication to extend their battery life. This results in well-defined communication patterns with possibly predictable intervals of activity that are easily detected and jammed by a statistical jammer. We present an anti-jamming protocol for sensor networks which eliminates spatio-temporal patterns of communication while maintaining coordinated and contention-free communication across the network. Our protocol, WisperNet, is time-synchronized and uses coordinated temporal randomization for slot schedules and slot durations at the link layer and adapts routes to avoid jammers in the network layer. Through analysis, simulation and experimentation 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. WisperNet has been implemented on the FireFly sensor network platform.

Keywords

Anti-Jamming, Wireless Networks

Disciplines

Electrical and Computer Engineering | Engineering

Comments

Suggested Citation:
Pajic, M. and R. Mangharam. “Anti-Jamming for Embedded Wireless Networks”. ACM International Conference on Information Processing in Sensor Networks (IPSN'09). April 2009.

Anti-Jamming for Embedded Wireless Networks

Miroslav Pajic and Rahul Mangharam
Department of Electrical and Systems Engineering
University of Pennsylvania
{pajic, rahulm}@seas.upenn.edu

ABSTRACT

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. In the context of energy-constrained wireless sensor networks, nodes are scheduled to maximize the common sleep duration and coordinate communication to extend their battery life. This results in well-defined communication patterns with possibly predictable intervals of activity that are easily detected and jammed by a statistical jammer. We present an anti-jamming protocol for sensor networks which eliminates spatio-temporal patterns of communication while maintaining coordinated and contention-free communication across the network. Our protocol, WisperNet, is time-synchronized and uses coordinated temporal randomization for slot schedules and slot durations at the link layer and adapts routes to avoid jammers in the network layer. Through analysis, simulation and experimentation 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. WisperNet has been implemented on the FireFly sensor network platform.

Categories and Subject Descriptors

C.2 [**Computer Systems Organization**]: Computer-Communication Networks—*Wireless communication*

General Terms

Security, Design, Algorithms

IPSN '09, San Francisco, April 13-16,2009 , San Francisco, CA, USA

Keywords

Anti-jamming, MAC protocol, wireless sensor networks

1. INTRODUCTION

Jamming is the radiation of electromagnetic energy in a communication channel which reduces the effective use of the electromagnetic spectrum for legitimate communication. Jamming results in a loss of link reliability, increased energy consumption, extended packet delays and disruption of end-to-end routes. Jamming may be both malicious with the intention to block communication of an adversary or non-malicious in the form of unintended channel interference. In the context of embedded wireless networks for time-critical and safety critical operation such 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*, is 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 re-locate or increase the cost of their links so the network may adapt its routes.

Methods for anti-jamming must therefore address threats due to both temporal patterns at the link layer and spatial distribution of routes in the network layer. Our goal in designing WisperNet, an anti-jamming protocol is to reduce or eliminate spatio-temporal patterns in communication while maintaining energy-efficient, coordinated and collision-free operation in multi-hop 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 inter-arrival 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 co-located 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 spatio-temporal patterns in communication are exposed to external observers.

In the context of multi-hop embedded wireless networks, which are battery-operated and require low-energy consumption, we require the following properties from the anti-jamming protocol:

1. Non-predictable schedules: Transmission instances (e.g. slot assignments) are randomized and non-repeating 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.

2. Non-predictable 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.

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.

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 anti-jamming 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.

2. BACKGROUND AND RELATED WORK

To understand the inherent tradeoff 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 [1] and [2], 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 a known minimum and maximum interval. 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 [3], a statistical jamming model is described where the jammer first observes temporal patterns in channel activity, extracts a histogram of inter-arrival 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 inter-arrival times and is thus able to maximize its censorship ratio with relatively little effort. Fig. 1(a) illustrates the relative censorship ratio and the energy-efficiency of the different jammers. Fig. 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.

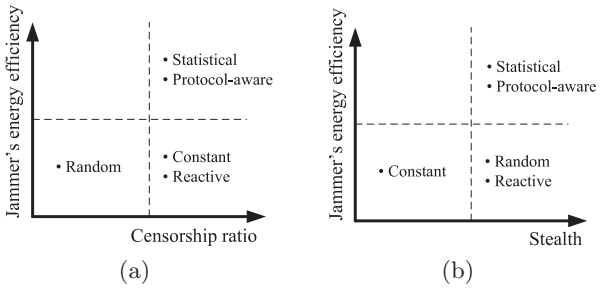


Figure 1: Jammer's Energy efficiency vs. (a) Censorship ratio (b) Stealth

2.1.2 Techniques for Robust Transmission

The traditional defenses against jamming include spread spectrum techniques[4] and frequency hopping at the physical layer. While these 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 pseudo-random noise code or frequency hopping sequence.

There have been several efforts to make communication in sensor networks more robust in the presence of a jammer. In [5], 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 [6].

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 [7] 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 its own position along with positions of all its neighbors. 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.

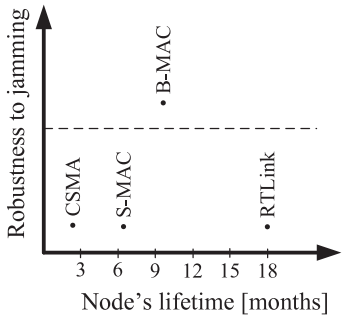


Figure 2: Comparison of robustness to jamming and energy efficiency of sensor MAC protocols

2.2.1 Energy-Efficient MAC Protocols

Several MAC protocols have been proposed for low-power operation for multi-hop wireless mesh networks. Such protocols may be categorized by their use of time synchronization as asynchronous [8], loosely synchronous [9, 10] and fully synchronized protocols [11, 12]. 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) [8] protocol performs excellent 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 0.25ms. 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 [9] and T-MAC [13] employ local sleep-wake schedules know 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 to transmissions. S-MAC results in clustering of channel activity and is hence vulnerable to a statistical jammer.

Synchronous protocols such as RT-Link [12], 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 50μs 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 (30W) 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 [12]. Fig. 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 10s event sample period (based on measurement values from [12], [9]). While RT-Link nodes communicate in periodic and well-defined fixed-size time slots, a statistical jammer is able to easily determine the channel activity schedule and duration of each scheduled transmission. 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 in explicit patterns in packet inter-arrival 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 3ms average transmission duration. In the case of S-MAC, we observe that all nodes quickly converge on one major activity period of 215ms. In Fig. 3, we also notice a spike close to 2ms. 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 Fig. 4. The other spikes with lower intensity are harmonics due to multiples of 32 slots in a frame. In both cases we observe distinct inter-arrival patterns which enable a statistical jammer to efficiently attack both protocols.

2.3 Assumptions

We make several assumptions in the design and evaluation of WisperNet. We assume the jammer is as energy-constrained as a legitimate node and must maintain a stealth operation with a low duty-cycle. All packets exchanged between nodes are encrypted with a group

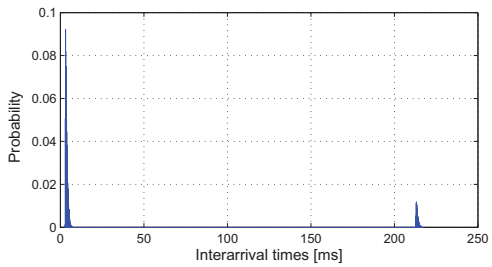


Figure 3: SMAC PDF for 15% utilization

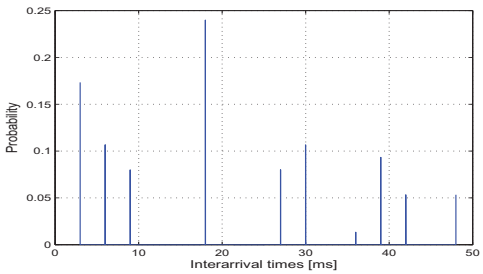


Figure 4: RT-Link PDF for 15% utilization

key shared by legitimate nodes and hence the jammer is not protocol-aware. We consider both malicious and non-malicious jamming and do not differentiate between them as the anti-jamming mechanisms are native to the link and network protocol. The transmission power is 0dBm (1mW) and in the worst case (with maximum power link jamming) the nominal packet delivery rate is never below $\tilde{2}0\%$. This has been demonstrated in previous experiments [12]. For simplicity, we presume that interference range is equal to the transmission range of one hop. This restriction does not limit our results. We assume all communication is between a central gateway and each of the nodes across one or more hops.

3. ANTI-JAMMING WITH COORDINATED SPATIO-TEMPORAL 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.

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 inter-arrival 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.

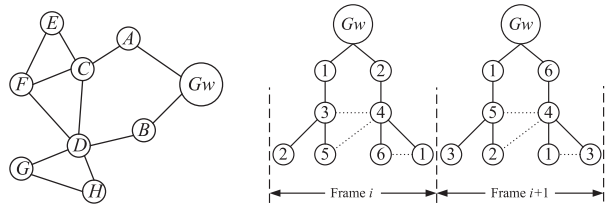
3.1 WisperNet-Time: Co-ordinated Temporal Randomization

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 [14] and using the basic synchronization mechanisms adopted in the RT-Link protocol. 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 sync. 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. Fig. 5(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 non-colliding schedules that can be changed for every frame, as shown in Fig. 5(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 Pseudo-Random 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.



(a) Physical network topology (b) Logical network topology

Figure 5: (a)Example network topology (b) collision-free, frame-to-frame transmit schedule

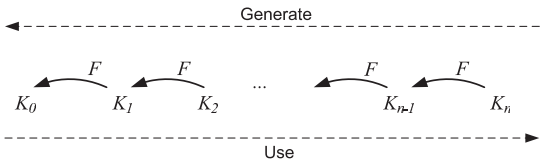


Figure 6: Generation of keys at the gateway, using a one-way hash function.

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.

To obtain non-repeating 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 Fig. 6):

$$K_j = F(K_{j+1}), j = 0, 1, 2, ...n-1.$$

As F is a one-way function, all previous members of chain $(K_0, K_1, ..., K_{j-1})$ can be calculated from some chain element K_j but subsequent chain members K_j, K_{j+1},...,K_n [15] cannot be derived. This authentication scheme is similar to [16, 17] but its use for scheduling is new.

We use the SHA1-HMAC[18] keyed-hash function to generate the current slot schedule. Therefore, for schedule computation $HMAC(ID, K_j)$ is used, where K_j presents 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 i-th frame ($i = 0, 1, 2, ...31$) is determined by i-th group of 5 bits. These 5 bits represent the randomly selected slot index in each of the 32 frames.

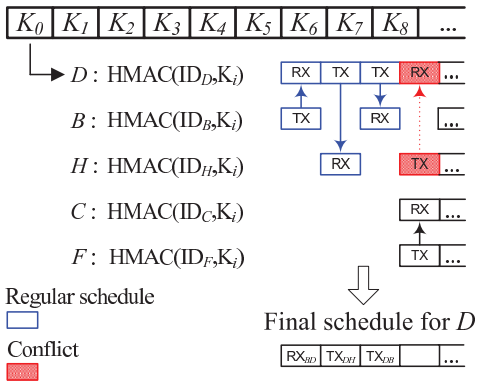


Figure 7: Implicit conflict resolution

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 i-th frame's transmission schedule is determined by $(32-i)$-th 5 bits group (i.e. reverse order of the transmission schedule). Since the number of frames is even, the transmission schedule and precedence are never extracted from same group of bits. Therefore, to compute its schedule in one sync period with 32 frames, 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 the node IDs of all k-hop neighbors are known (when node joins the network it broadcast 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 7. After schedule conflicts are resolved implicitly based on the higher precedence, the node follows the combined transmit and receive schedule in a single vector.

3.1.2 *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 inter-arrival times [3]. 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 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

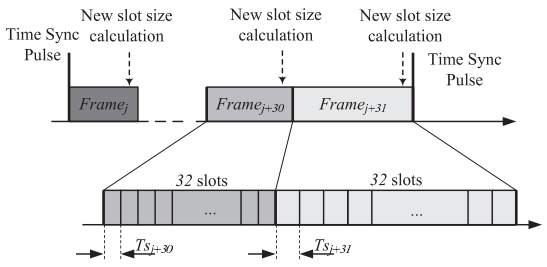


Figure 8: Slot size randomization on a frame-by-frame basis

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 output, which is also calculated on a frame-by-frame basis, defines the slot size for the next frame as shown in Fig. 8. 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 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 inter-arrivals times. The goal of our anti-jamming 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 inter-arrival 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 vs. 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 [12], 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 [1 5]ms. A maximum slot size of 5ms is chosen to match the maximum message size of 128 bytes for IEEE 802.15.4 transceiver with data rate of 250kbps[6]. 128 bytes can be sent with transmission duration of 4.2ms and the rest of the slot time is used for inter-slot processing and for guard times. A simulation for 10000 sync 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 inter-arrival 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 [3]. Results for PDF of inter-arrival times are shown in in Fig. 9 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 signficantly 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. With U below 50%, a small increment can be

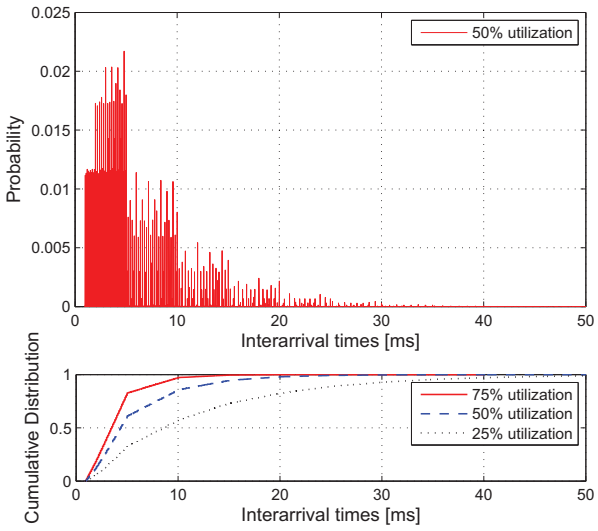


Figure 9: (Top) PDF of inter-arrival times for WisperNet. (Bottom) Corresponding CDF.

seen on spikes in (5 10]ms interval. Also with integer multiples of some slot sizes within [1 5]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.

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 periodically receives fixed size data blocks with a smaller size than for an average slot size (e.g. 3ms).

Fig. 10 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 $3ms$, the delay at the last node would be $20 \cdot 3ms = 60ms$. Here average delay is around 75ms, but with 95% probability interval [67 88]ms. This shows that randomization introduces some variability into the communication end-to-end delay estimation.

The first node requires a buffer with an additional 512 bytes of memory for data queuing while all other require an additional buffer for one maximum sized packet.

3.2.2 Comparative Analysis of MAC Protocols

Fig. 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 $150\mu s$-long pulse

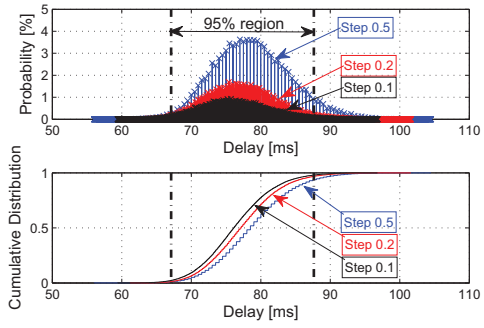


Figure 10: Message delay and its Cumulative Distribution at last node, for 20 nodes chain

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 frame, where every slot is 3ms 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 1ms, 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 pre-stored 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 anti-jamming. 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) $w_j(u, v), (j = 1, 2, ..., k)$ are associated. For a tree T in graph G, the aggregate weight $W_j(T)$ is defined as

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

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 $j = 1, 2, ...k$. 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) [19]. If $L \neq V$ and also only one weighting function is considered, problem is equivalent to Steiner minimal

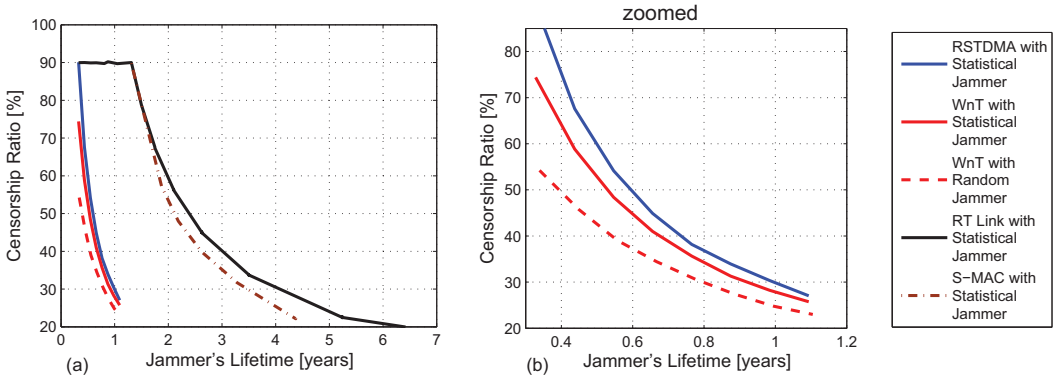


Figure 11: (a) Dependency between jammer's lifetime and Censorship Ratio and (b) zoomed

tree problem (SMT). SMT is a NP-complete problem, but several heuristics exist which resolve SMT problem in both a centralized and a distributed manner.

For WisperNet-Space we only considered network 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 [2]. 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 $\prod_{e \in P} PDR(e)$. Our goal is to achieve maximum reliability on a path P, which is equivalent to maximizing

$$ln \prod_{e \in P} PDR(e) = \sum_{e \in P} lnPDR(e).$$

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

$$w_r(u, v) = |lnPDR(u, v)|.$$

4.1 Active topology update

For adaptive routing in WisperNet-Space, we use a 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 can not 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:

$$w_r(u, v) = \begin{cases} |lnPDR(u, v)|, e \in T \\ \rho \cdot w_r(u, v), e \notin T \end{cases}$$

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 very 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) [19], 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 look-up table with size M is sent, where i-th 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.

4.2 Topology Maintenance and Updates

WisperNet-Space computes a new network topology every 128 cycles. Given the average slot size of 3ms 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, are considered *inactive nodes*. The key challenge during a topology update is to activate the inactive nodes, which 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.

Algorithm 1 describes the gateway's procedure for topology dissemination. The generated message with the information about new network topology is distributed over the network using all active links. Since some currently inactive nodes may be part of next active topology, the mechanism to inform them is included.

Algorithm 1 Gateway procedure description

```
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
```

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 nodes schedule its transmission of the new configuration to neighboring node and keep 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.

Since 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 the first 10% of these slots. Given this, we allowed the last 20% of these slots (i.e. configuration slots 820-1024) to be used as contention slots for addmission of new nodes only. These slots enable nodes that want to join the network to announce their presence to neighboring nodes (via a HELLO packet in RT-Link), so that they can be initially included as inactive nodes in the network.

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 distributed nodes in a $4km$ x $4km$ 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).

Fig. 12(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 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 realible paths and it can not go below this minimum value. The best configuration (Fig. 13) includes 0 links jammed in both direction and 2 links jammed in only one direction.

We observed that at some intermediate moments (for example moments t_1 and t_2 as seen in Fig. 12(b) and cor-

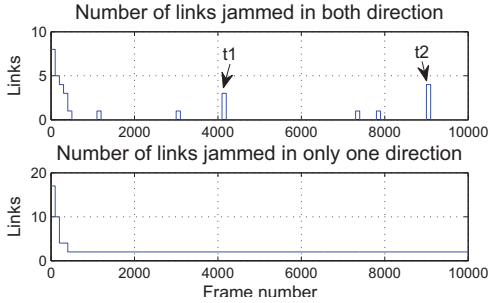
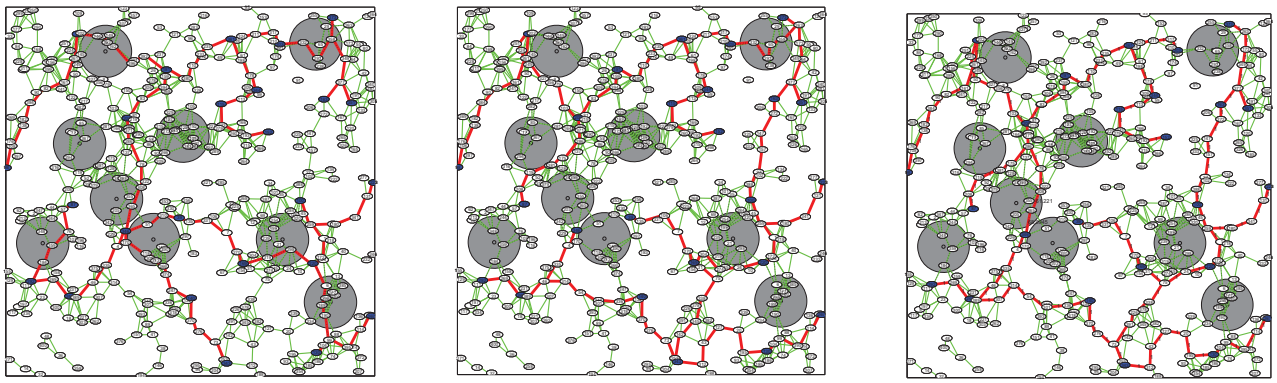


Figure 13: Number of links jammed in 2/1 direction used for communication



(a) Network topology - beginning (b) At an intermediate time t_1 (c) Network topology - best

Figure 12: 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

responding Fig. 14) network routes with more jammed links were chosen, which directly resulted in 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 Fig. 12(b), three links are jammed in both directions in the top-right corner. 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 the currently used subgraph. We observe these spikes in the network's censorship ratio in Fig. 14. Fig. 12(c) presents the best 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 observer that the number of active links does not vary much. We also noticed the stretch factor of the network path lengths was always $\leq$1.3.

5. IMPLEMENTATION AND EVALUATION

The WisperNet anti-jamming protocol was implemented on a network of FireFly sensor nodes[14]. Each node consists of a microcontroller, an IEEE 802.15.4, 2.4 GHz transceiver and multiple sensors. FireFly nodes have 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) plugged into the wall-outlet and used the building's power grid as an extended AM antenna and was thus able to cover the entire building. Nodes were synchronized with a 50μs pulse that was transmitted every 10 seconds from the AM transmitter (see [14] for details). The AM pulse has a jitter of $\leq 150\mu$s.

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 16MHz TI MSP430F22x microcontroller. Each node ran the nano-RK[20] real-time operating system and the RT-Link[12] link protocol. The RT-Link TDMA cycle includes 32 frames which in turn are composed of 32 slots. The slot size were assigned values from [1 5]ms. In our tests, every node attempts to transmit one message per frame.

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 12.5ms 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 337.5ms of CPU time used for these calculations in every sync period, where one sync period contains $32 \cdot 32 = 1024$ slots,

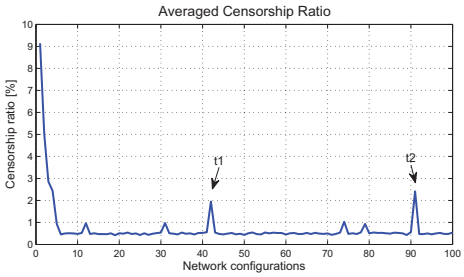


Figure 14: Averaged network's censorship ratio on 100 blocks, for implemented Steiner tree

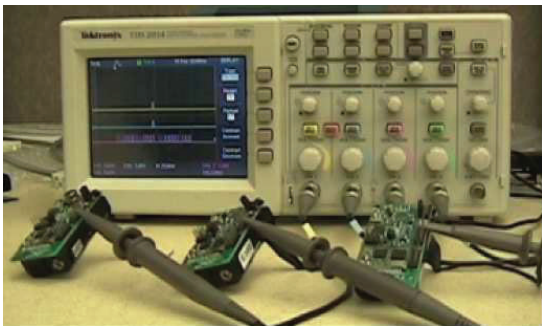


Figure 15: Experimental setup with 3 nodes

with sizes from 1ms to 5ms. Therefore in the worst case less than 33% of CPU processing is used for these calculations, while on average less than 11% is used. The implemented schedule randomization procedure uses 276 bytes of flash memory and 400 bytes of RAM. As only two nodes' schedules are required at once, this requires only 236 bytes for additional code size with 40 bytes of data pre-initialized in flash.

In Fig. 15, 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 Fig. 16, 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.

5.1 Limitations

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. These schemes are not conducive to energy-constrained and memory-constrained sensor networks. We aim to explore distributed solutions further.

6. CONCLUSION

In this paper we proposed the WisperNet anti-jamming protocol which uses 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).

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 $\leq$2% censorship rate for the network topologies explored in this work. The schedules derived from WisperNet are non-repeating, 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.

7. REFERENCES

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

[2] W. Xu, T. Wood, W. Trappe, and Y. Zhang. Channel Surfing and Spatial Retreats: Defenses against Wireless Denial of Service. In *ACM WiSe*, pages 80–89, 2004.

[3] Y. W. Law et al. Energy-efficient link-layer jamming attacks. In *ACM SASN*, 2005.

[4] A. J. Viterbi. Spread Spectrum Communications: Myths and Realities. In *IEEE Comm. Magazine*, 2002.

[5] A. D. Wood, J. A. Stankovic, and G. Zhou. DEEJAM: Defeating Energy-Efficient Jamming. *IEEE SECON*, 2007.

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

[7] A. D. Wood, J. A. Stankovic, and S. H. Son. JAM: A Jammed Area Mapping for Sensor Networks. In *IEEE RTSS*, 2003.

[8] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. *SenSys*, November 2005.

[9] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. *IEEE INFOCOM*, 2002.

[10] A. El-Hoiydi and J. Decotignie. WiseMac: An Ultra Low Power MAC Protocol. *ISCC*, 2004.

[11] L.F.W. van Hoesel and P.J.M. Havinga. A lightweight medium access protocol for wireless sensor networks. *INSS*, 2004.

[12] A. Rowe, R. Mangharam, and R. Rajkumar. RT-Link: A Time-Synchronized Link Protocol for Energy Constrained Multi-hop Wireless Networks. *IEEE SECON*, 2006.

[13] T. Dam and K. Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. *SenSys*, November 2003.

[14] R. Mangharam, A. Rowe, and R. Rajkumar. FireFly: A Cross-layer Platform for Real-time Embedded Wireless Networks. *Real-Time System Journal*, 37(3):183–231, 2007.

[15] B. Schneier. *Applied Cryptography*. John Willey Inc., 1996.

[16] A. Perrig and et. al. SPINS: Security Protocols for Sensor Networks. *Wireless Networks*, 8(5):521–534, 2002.

[17] A. Perrig, R. Canetti, D. Tygar, and D. Song. The TESLA Broadcast Authentication Protocol. *Cryptobytes*, 5(2), 2002.

[18] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. *RFC 2104*, 1997.

[19] B. Y. Wu and Kun-Mao Chao. *Spanning Trees and Optimization Problems*. Chapman and Hall, CRC Press, 2004.

[20] A. Eswaran, A. Rowe, and R. Rajkumar. Nano-RK: Energy aware Resource centric RTOS. *IEEE RTSS*, 2005.

Figure 16: WisperNet-Time on Oscilloscope