Channel Surfing: Defending Wireless Sensor Networks from Interference

Wenyuan Xu
WINLAB
Rutgers University
North Brunswick, NJ 08902
wenyuan@winlab.rutgers.edu

Wade Trappe
WINLAB
Rutgers University
North Brunswick, NJ 08902
trappe@winlab.rutgers.edu

Yanyong Zhang
WINLAB
Rutgers University
North Brunswick, NJ 08902
yyzhang@winlab.rutgers.edu

ABSTRACT

Wireless sensor networks are susceptible to interference that can disrupt sensor communication. In order to cope with this disruption, we explore channel surfing, whereby the sensor nodes adapt their channel assignments to restore network connectivity in the presence of interference. We explore two different approaches to channel surfing: coordinated channel switching, where the entire sensor network adjusts its channel; and spectral multiplexing, where nodes in a jammed region switch channels while nodes on the boundary of a jammed region act as radio relays between different spectral zones. For spectral multiplexing, we have devised both synchronous and asynchronous strategies to facilitate the spectral scheduling needed to improve network fidelity when sensor nodes operate on multiple channels. In designing these algorithms, we have taken a system-oriented approach that has focused on exploring actual implementation issues under realistic network settings. We have implemented these proposed methods on a testbed of 30 Mica2 sensor nodes, and the experimental results show that these strategies can each repair network connectivity in the presence of interference without introducing significant overhead.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*

General Terms

Security, Reliability, Experimentation

Keywords

Jamming, Radio Interference, Channel Surfing

1. INTRODUCTION

As wireless networks become increasingly pervasive, it is very likely that the radio environment will not be favorable. Notably, as an increasing number of wireless devices are deployed that use "open spectrum", it will be inevitable that there will be problems of spectrum coexistence. This is the well-known near-far problem of communications, which manifests itself in dense urban environments where cordless phones suffer degraded performance when many devices operate simultaneously within a small distance of each other. Interference problems can be projected for sensor networks as more sensor devices are deployed. Another reason stems from the fact that most sensor networks will consist of commodity devices that can be easily purchased and reprogrammed to interfere with communications.

Whether intentional or not, interference/jamming will be a serious threat to the availability of sensor services. The traditional approach to coping with radio jamming is to employ sophisticated physical-layer technologies (e.g. spread spectrum). Such methods imply more expensive transceivers and most commodity sensor and wireless networks do not employ sufficiently strong spreading to survive jamming. Instead, systems like the Berkeley Mica2, Zigbee and 802.11 are based upon carrier-sensing for medium access, and hence are particularly susceptible to radio interference. Recent studies [1,2], have revealed the relative ease with which jamming can be conducted on such networks.

In this paper, we examine the ability of a sensor network to cope with radio interference. We propose the use of *channel surfing*, whereby the sensor nodes adapt their frequency allocations as needed to avoid interference. The challenging research question here is how to establish network connectivity between multiple frequency zones. The inherent diversity in network configuration, interference model, and platform setup suggests that no single solution is sufficient. We examine three strategies to restore network connectivity across multiple channels, each having unique characteristics and advantages. Although channel surfing may be applied to more general wireless networks (e.g. 802.11), in order to validate our strategies, we focused our study on a sensor network platform, and have implemented our methods on a 30-node Mica2 sensor network testbed. During the process of implementation, we have overcome a number of challenges and have demonstrated that all three strategies can maintain network operations in the presence of jamming/interference.

We begin the paper in Section 2 by providing an overview of the sensor network and interference model used in our studies. We next introduce the channel surfing in Section 3 and Section 4, where we detail a set of increasingly sophisticated channel surfing protocols. In Section 5 and Section 6, we describe our validation effort on our sensor testbed. We wrap up the paper by discussing related work in Section 7, and provide concluding remarks in Section 8.

IPSN'07, April 25-27, 2007, Cambridge, Massachusetts, USA.

2. SYSTEM MODELS

We now outline the basic sensor communication and jamming model that we use throughout this paper.

Our Sensor Communication Paradigm: Channel surfing requires that sensor radios change their *channel* allocations. Thus, the radios employed must have a notion of a channel. Most sensors have a natural form of channelization that is accomplished by changing the carrier frequency. For example, in our validation, we use the 916.7MHz Mica2 platform and separate the channels by 800KHz.

Another important factor in the sensor communication paradigm is the choice of the data dissemination method and the associated routing protocol. In this work, we have chosen to focus on the many-to-one model in which many sensors report their findings to one or a few sinks. Specifically, our studies focus on tree-based routing schemes [3], whereby a routing tree is formed with the sink node serving as the root of the tree. A node selects its routing *parent* as its best radio neighbor in the direction of the tree's root. A node usually has a parent (except the sink) and one or more children (except the leaf nodes), whereby it receives data packets from its children, and sends packets to its parent. A node's parent and children are considered its *neighbors*. Besides the parent and children, there may be other nodes that are within a node's communication range. These nodes are *not* neighbors of the node. In this paper, we use the term neighbor to refer to the topology-based relationship rather than a physical location-based relationship.

Our Interference Model: When considering issues of radio interference and jamming, it must be emphasized that there is a very broad range of capabilities that one might assume is available to the interferer, ranging from whether the interferer is incidental or intentional, powerful or resource-constrained, narrowband or broadband, or static or adaptive. It is immediately apparent that if the jammer is a high-powered, broadband source of interference (e.g. capable of occupying all channels simultaneously), then there is no hope for building a resilient sensor network short of choosing a different PHY-layer transceiver with a powerful anti-jam margin [4,5]. Further, it should also be noted that an aggressive adversary may jam a single channel (e.g. by reprogramming another sensor) at a time and rapidly switch between channels to effectively disrupt network services across all channels. Both cases represent powerful, aggressive broadband jamming adversaries.

Instead of considering powerful interference models for which the only viable defense might be powerful physical layer techniques, in this work, we consider a non-intentional or a relatively benign adversary in which the interferer blocks one (or even a few) of the channels at a time. This model been considered elsewhere in the literature [1, 2, 6, 7]. Further, even if the interferer hops to different channels, we assume that the interferer stays on one channel for a brief period of time before switching to another channel. Related to these assumptions, we note that we consider traditional security threats (such as authentication, communication confidentiality, and coping with node compromise) to be orthogonal to the issues discussed in this paper.

A jammer, whether incidental or intentional, can significantly affect the reliability of the network's communications. As the starting point for coping with jamming, it is necessary to detect jamming. In this paper, we utilize our detection scheme of [2], which involves a consistency checking process on each node to ensure reliable identification of jamming attacks.

3. CHANNEL SURFING OVERVIEW

In channel surfing, those nodes that detect themselves as *jammed* nodes should immediately switch to another orthogonal channel and wait for opportunities to reconnect to the rest of the network. After the jammed nodes lose connectivity, their neighbors, which we refer to as *boundary* nodes, will discover the disappearance of their jammed neighbor nodes and temporally switch to the new channel to search for them. If the lost neighbors are found on the new channel, the boundary nodes will participate in rebuilding the connectivity of the entire network.

Channel surfing imposes several challenges to the underlying design. The first challenge concerns the potential boundary nodes. The basic scheme specifies that a boundary node should switch to a new channel after its neighbors are jammed and have escaped to the new channel. However, if a node immediately probes the next channel whenever it experiences a poor link quality with any of its neighbors, the system will enter a non-stable state because wireless sensor networks inherently experience frequent link quality degradations or even topological changes [8,9]. Fortunately, after carefully studying the working of the underlying system, we found that it is possible for boundary nodes to correctly differentiate jammed neighbors from those neighbors that just had a poor link with the boundary node. This can be explained as follows. For the tree-based routing, a node has precisely two types of neighbors: a parent and several children. If a node has a poor connection with its parent, it will first attempt to find another suitable parent node. Only if there is no suitable replacement parent will the node probe the next channel. In this way, the node does not need to switch channels if it can still maintain its normal network operations. In the case of losing a node's children, if the lost child node was not jammed, but just connected to a new parent, the node *should* hear its former child's routing announcement. If it does not witness the child's routing announcement within a specified window of time, then it will probe the next channel looking for its lost child.

After detecting the loss of a neighbor, the boundary node should not switch to the new channel too quickly, it may arrive at the new channel before the jammed nodes. To understand this, consider a scenario where the jammer starts interference at time t_0. At that time, the jammed nodes will not be able to send out packets. However, since it takes less time for a node to detect the absence of a neighbor than it does for a node to decide it is jammed, the boundary nodes will detect the absence of a jammed node at time $t_0 + \delta_1$, while the jammed node will declare itself jammed at $t_0 + \delta_2$, where $\delta_2 > \delta_1$. If the boundary nodes switch to the new channel immediately after $t_0 + \delta_1$, they will not find the jammed nodes there. Rather than have the boundary node wait on the next channel, which would prevent it from conducting its primary objective of relaying messages to the sink, or have the node constantly flip-flop between channels looking for its children, we should make the boundary nodes wait for at least an additional $\delta_2 - \delta_1$ amount of time before switching to the next channel. The values of δ_1 and δ_2 are characteristic of the particular routing protocol, and the jamming detection scheme.

In addition to the switch timing for a boundary node, it is

important to have a discovery protocol by which a boundary node can find its neighbors on the new channel. After a node switches to the new channel searching for its neighbors, it should send out an "inquiry" message, such as "Is my neighbor X here?" If it receives a reply from X, it will start working on repairing the connectivity between X and the sink. Otherwise, it waits for time δ to send another message. If the node does not hear from X after a few trials, it assumes the child is not jammed, returns to the original channel and resumes its original operation in the network. In total, the time spent probing the next channel should be less than δ_2 to avoid a cascading of channel probing.

It is desirable to choose the next channel so that the adversary cannot predict what channel the nodes will surf to. We may choose to chain the channel selections using a keyed pseudo-random generator. If the n-th channel assignment is $C(n)$, then we take $C(n+1) = E_K(C(n))$, where K is a key shared by all nodes in the network that is used exclusively for channel assignment. If ever $C(n+1) = C(n)$, then the channel assignment proceeds to $C(n+2)$ and so on until a different channel is selected. Finally, if the jammer can block several channels, then after a jammed node escapes to a new channel, it should first detect whether the channel is jammed before it starts working on that channel.

4. CHANNEL SURFING STRATEGIES

After the boundary nodes discover that their neighbors are jammed and have escaped to another channel, they will attempt to reconnect the jammed nodes with the rest of the network. We propose two different classes of techniques that the boundary nodes can use to repair network connectivity: (1) Coordinated Channel Switching, in which the boundary nodes participate in transitioning the entire network to the new channel to rebuild total network connectivity on the new channel; and (2) Spectral Multiplexing, where boundary nodes multiplex between the old channel and the new channel, serving as a "bridge" that connects nodes operating on different channels.

4.1 Coordinated Channel Switching

In coordinated channel switching, the entire network must coordinate its evasion of the interference by switching to the next channel and resuming network operation there. The strategy involves by a transition phase during which an increasing amount of nodes switch to the next channel. Following the transition, the entire network resumes stable operation on the next channel. The scheme begins with the jammed nodes detecting they are jammed. After a set amount of time, the boundary nodes will notice that they have not received messages from their jammed neighbors. The boundary nodes will then probe the next interference-free channel, searching for their lost neighbors. If the boundary node finds a neighbor residing on the next channel, it will switch back to the old channel, broadcast a *channel switch command*, and return to the new channel. Once a node receives this notice, it rebroadcasts the command and switches to the channel specified.

Algorithm Walk-through: In order to illustrate the Coordinated Channel Switching algorithm, let us walk through the example depicted in Figures 1(a)-(d). Here, the jammer X affects nodes $\{D, I, J, O\}$. Upon detecting they are jammed, these four nodes switch to channel 2, as shown in Figure 1(a). The dashed-dot lines indicate links that exist in

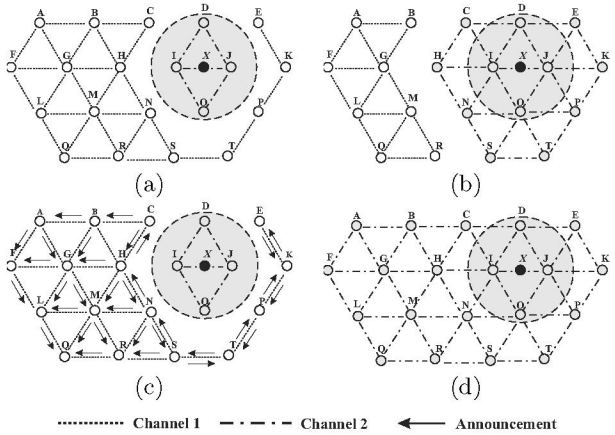


Figure 1: A walk-through of coordinated channel switching. The shaded area depicts the jammed area, with the jammer X at the center.

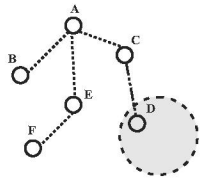
channel 2, while the dotted lines correspond to links in the first channel. The boundary nodes $\{C, E, H, K, N, P, S, T\}$ will notice that their jammed neighbors are no longer on channel 1, and will probe channel 2. After finding the jammed nodes on the new channel, the boundary nodes return to the original channel temporarily and broadcast a switching notice (which contains the ID of the sender node, and the channel to switch to) to the rest of the network, propagating through the channel 1 subnetwork, as shown in Figure 1(c). The boundary nodes join the jammed nodes on the new channel after broadcasting the notice. Shortly thereafter, the rest of the nodes will switch to the new channel after receiving the switching notice, and reestablish the network on channel 2, as shown in Figure 1(d).

Algorithm Challenges: The major challenge facing this scheme is the fact that unreliable links can cause some nodes to miss a channel switch notice. However, the switch command is typically broadcasted independently by multiple boundary nodes. Thus a node is very likely to receive at least one notice. Even should a node not receive a switch notice, it will still autonomously move to the next channel as it will detect that it cannot receive messages from neighbors that have already switched to the new channel.

Discussion: One advantage of this scheme is its simplicity. Further, the success of performing a coordinated channel switch doesn't depend on the likelihood that each individual node can detect the loss of its neighbors but, rather, as long as one of the boundary nodes finds its lost neighbors in the new channel and informs the rest of network, the network will resume its connectivity in the new channel in spite of the radio interference. Finally, we note that the broadcasted channel switch command should be authenticated [10] to prevent malicious message injection by the adversary.

4.2 Spectral Multiplexing

Performing a coordinated channel switch requires the entire network to reestablish the routing tree as the link connectivity will not be the same on the new channel. The global nature of coordinated channel switching can be a source for significant network cost, and a natural alternative is to employ a local response where only jammed nodes switch channels, while non-jammed nodes remain on the original channel. To guarantee the communication between



(a) the network topology

slot 1	A, B, C, E, F
slot 2	C, D

(b) the global schedule

Figure 2: The synchronous spectral multiplexing algorithm.

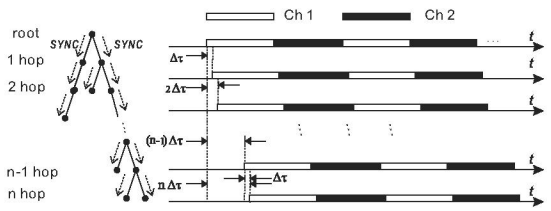


Figure 3: The synchronization mechanism for the synchronous spectral multiplexing strategy.

these two frequency zones, boundary nodes have to work on both channels by repeatedly switching back and forth between two channels to *relay* packets, a process we call *spectral multiplexing*.

For spectral multiplexing, the primary challenge lies in the fact that the boundary nodes must carefully decide when they should be on which channel so that they can minimize the number of packets not delivered due to the sender-receiver frequency mismatch. If the boundary nodes are configured with dual radios, as suggested in [6], this scheduling is unnecessary. However, commercial sensor platforms only have one radio interface. It is thus crucial to ensure that the sender and the receiver are able to work on the same channel when they want to exchange messages. Synchronization is thus needed to coordinate the spectral schedules of the sender and the receiver. The overall scheduling objective is to ensure that multiplexing nodes are on the right channel when the neighbors on that channel are ready to transmit[1].

In general, there are two ways of coordinating schedules from different entities: one is to have all the entities adopt synchronous schedules, and the other is to operate in an asynchronous fashion. In synchronous multiplexing all nodes share the same schedule by dividing the global time axis into different slots and assigning one slot to a channel; while in asynchronous multiplexing, each node operates on a local schedule, and boundary nodes make local decisions about when to switch channel.

4.2.1 Synchronous Spectral Multiplexing

In synchronous spectral multiplexing, the entire network is governed by one global clock [12]. The global time axis is divided into slots, and multiple slots form a round. The number of slots in a round is determined by the number of channels the network is allowed to operate on at any specific time. (In this paper, we limit our discussions on situations where the network works on 2 channels simultaneously, and the discussion can be easily extended to cover situations with more than 2 channels.) Each slot is assigned to a single channel, and during that time slot, network nodes may only use the corresponding channel– regardless of whether they are jammed, boundary nodes, or not. At the end of a time slot, the entire network utilizes the next channel and, again, the nodes that are not using the next channel do not transmit, nor must they switch channels unless they are multiplexing boundary nodes. By following this global schedule, we can avoid frequency mismatch between a pair of communicators.

Algorithm Walk-through: Figure 2(a) presents an example network scenario in which D is jammed, and switches to channel 2. Its parent node, C, thus becomes a boundary node, and has to multiplex between two channels. The rest of the nodes continue to work on channel 1. The global schedule for this case is shown in Figure 2(b), which has two slots for each round, with slot 1 allocated to channel 1, and slot 2 to channel 2. Following this schedule, during slot 1, nodes $\{A, B, E, F\}$ work as normal. Node C sends out packets to its parent A, but does not receive any packets from D. At the end of slot 1, these nodes stop their activities on channel 1, and node C switches to channel 2. During slot 2, the only transmitting node is D, and C buffers all the packets it receives from D. At the end of slot 2, D ends its transmissions and C switches to channel 1. These two slots keep alternating in this fashion until the radio interference ends, or for the lifetime of the network.

Algorithm Challenges: There are several practical challenges associated with this scheme.

Synchronization: In order to efficiently synchronize node schedules, instead of synchronizing the physical clock of each network node, in our implementation we let each node adopt a timer to demarcate slots and synchronize these timers. In addition, instead of employing traditional pair-wise synchronization, the root initiates the synchronization process by broadcasting SYNC packets to its children, and the children broadcast SYNC to their children, and so on as depicted in Figure 3. This simple protocol can effectively minimize the synchronization error between a pair of neighbors to $\triangle\tau$, which only includes delays involved in sending, propagating, and receiving $SYNC$ packets. To avoid the timer drift on different nodes, $SYNC$ packets are sent periodically at an interval much larger than the slot duration. This process is made more complicated because synchronizing parties may not be on the same channel at the time of synchronization. To ensure that nodes on both channels are synchronized, the boundary nodes should send these $SYNC$ packets in rapid succession across both channels.

Initiation: As soon as a boundary node discovers that jammed nodes have evaded to the new channel, it will send a message to the root on the original channel that contains a list of the channels it will be working on. Eventually, the root will have the full list of channels the network has to operate on, and it will create a slotted channel schedule based on this list, and broadcast the schedule down the tree, along with the clock synchronization packets.

Slot Duration: Slot duration is an important parameter in the synchronous spectral multiplexing algorithm. At first glance, it seems intuitive that a shorter slot duration is more desirable because, if a node stays on one channel short enough, the required buffer space will be smaller and, more importantly, less latency would be incurred. However, we found that a smaller slot can be problematic as well, mainly due to the overhead associated with switching channels. Before a node switches to a new channel, it has to

[1]The need for scheduling transmissions has also been considered in the context of duty cycling, as in S-MAC [11], in order to preserve energy.

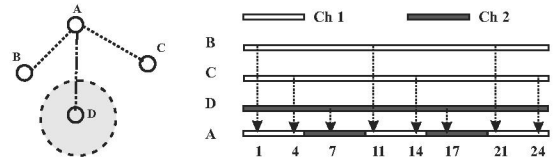
complete receiving all the packets that are in transmission. In order to guarantee this, after the Timer expires, we let each node wait for a small amount of time for all the possible transmissions to complete. In our implementation, this translates into the parent node waiting a little longer than child nodes because the receiving side is usually the parent node in tree-based routing. Another problem with short slot durations is that proportionately the synchronization errors will be relatively large with respect to the duration of a slot, thereby affecting this scheme's efficiency. Finally, we note that there is a radio startup cost associated with switching channels (e.g. 250msec for the CC1000 radio chip in the Mica2 mote).

In this study, we choose to adopt the largest slot durations that can satisfy the available buffer space constraints, and we consider our underlying sensing application model to have a periodic traffic pattern. Further, we have empirically witnessed that boundary nodes are more likely parent nodes of the jammed nodes (children of jammed nodes typically look for alternate parents on the original channel), and thus boundary nodes will merely receive on channel 2, but will both receive and send on channel 1. Specifically, based on the traffic rate from each channel and the buffer size, each boundary node calculates the longest stay time it can have on each channel (usually a node should stay on each channel for the same amount of time). In order to understand the calculation, let us look at an example. Suppose a boundary node A has a buffer that can support 10 slots, and it has 1 child on channel 1 that produces 20 packets per second, and 2 children on channel 2 each of which produces 10 packets per second. A can at most stay on each channel for 250 msecs. By spending 250 msecs on each channel, it will receive 10 packets in a round (5 from each channel), which will fill up its buffer. After each boundary node independently calculates a slot duration, the sink will collect all the information, chooses the smallest one as the global slot duration and announces this.

Discussion: Synchronous multiplexing adopts a deterministic global schedule that governs the channel assignment of every node in the network. The deterministic nature of this algorithm guarantees that it can work well even under complex scenarios where multiple nodes need to work on multiple channels and these nodes are neighbors of each other. However, in order to achieve this, every node in the network must pay the extra overhead needed to maintain synchrony.

4.2.2 Asynchronous Multiplexing

In the asynchronous multiplexing algorithm, a node is only aware of its neighbors' channel information, but not the channel information of a remote node. The simplest spectral scheduling method is to have a boundary node flip its radio frequency between two channels in a round-robin fashion. However, a completely random round-robin multiplexing strategy ignores the schedules of the communicating parties, and would thus fare poorly. For example, suppose a jammed node, working on the new channel, sends packets at times 10, 20, 30, 40, and 50. If the corresponding boundary node stays on the new channel during time windows [1, 6], [13, 18], [25, 30], [37, 42], [49, 54], then it will miss the packets sent at 10, 20, and 30. The resulting packet loss ratio for the jammed node is now as high as 60%. The above example illustrates the limitation of a random round-robin



(a) the topology (b) the round-robin schedule

Figure 4: Illustration of the round-robin asynchronous spectral multiplexing algorithm.

scheme and highlights the need for some level of coordination between the boundary node and its neighbors for the asynchronous multiplexing scheme.

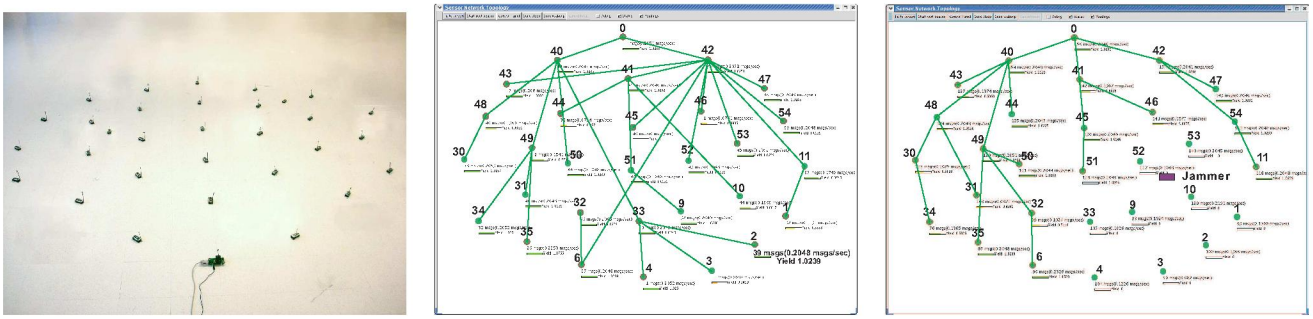
Algorithm Walk-through: Figure 4 illustrates the idea behind the asynchronous multiplexing scheme. In this example, the boundary node A has to receive packets from three nodes, B, C, and D, with the first two working on channel 1 and the last one working on channel 2. Suppose all three nodes send packets every 10 time units, starting at time 1, 4, and 7 respectively. In this case, starting from time 0, A decides to stay on one channel for 5 time units and then switches to the next channel for 5 time units. In this way, A can receive every packet from its neighbors.

Algorithm Challenges: The challenges associated with this schemes include synchronization and slot duration.

Synchronization: To coordinate the schedules of a boundary node and its children, we have adopted a simple protocol that involves the boundary node announcing its schedule (the duration it will stay in the new channel) by notifying its children just after it switches to a new channel. In the example in Figure 4(a), A notifies nodes B and C of its schedule as soon as it switches to channel 1, so that they can start transmissions when A enters channel 1, and stop transmissions after A leaves channel 1. Similarly, it also notifies D whenever it is on channel 2. We note that a child node must buffer both its own packets as well as packets coming from its own children while waiting for the dual-mode parent to return to the channel it is working on. To counteract the possibility that the notifications could be lost, a child should start to send its buffered packets immediately after it hears from its parent.

Slot Duration: Determining the slot duration in the asynchronous spectral multiplexing is easier than in the case of the synchronous spectral multiplexing, because in the former situation, not only can different boundary nodes employ different schedules, but they can also stay for a different amount time on each channel. For example, considering the boundary node usually is the parent of the jammed nodes, the boundary node should stay longer on the channel 1 than channel 2, as it has to forward all the packets received in both channels to its parent via channel 1.

Due to the nature of asynchronous spectral multiplexing, nodes can determine their slot durations in a more flexible fashion. Suppose a boundary node decides to stay on channel 1 for t_1 time and channel 2 for t_2 time ($t_1 \geq t_2$), where t_1 and t_2 are chosen according to the traffic volume on each channel and its buffer size. For example, it can choose to have $\frac{t_1}{r_1} + \frac{t_2}{r_2} = B$ where r_1 and r_2 are the traffic rates on the two channels respectively, and B is the buffer size. After setting this baseline schedule, the boundary node can adapt its switching rate as a response to varying network conditions (e.g. topology change, traffic rate change, etc).



(a) (b) (c)

Figure 5: Our Mica2 testbed is shown in (a). It consists of 30 motes that are placed on the floor. The sink is located at the bottom of the figure, with a programming board attached to it. The network topology is captured and presented in (b) and (c), with the former showing the topology prior to introducing the jammer, and the latter showing the topology shortly after the jammer is introduced.

Discussion: Compared to synchronous multiplexing, asynchronous multiplexing does not maintain a global schedule, and thus incurs less synchronization overhead. The advantage of asynchronous multiplexing is more pronounced when the jammed region is small and regular. For larger jammed areas, we will have more boundary nodes working on multiple channels, and thus the overhead gap between synchronous and asynchronous techniques lessens. A final advantage of the asynchronous method its ability to adapt to local traffic and buffer conditions.

5. SENSOR TESTBED AND METRICS

We now focus on our experimental validation efforts, and note some practical issues we faced.

5.1 Testbed Configuration

We have built our sensor network testbed using 30 Mica2 sensor motes. These devices each have a $902 - 928$ MHz Chipcon CC1000 radio. We used 916.7MHz as the original channel and separated our channels by 800KHz, effectively giving us 32 channels. The operating system running on each mote was TinyOS version 1.1.7 [13]. We attached one of the motes to a MIB510CA programming board in order to act as the network sink. In order to conduct experiments that exhibit repeatable characteristics, we chose an indoor laboratory area where we could fix the deployment across the experiments, as illustrated in Figure 5 (a). Due to space limitations, we reduced the radio range of each mote to roughly 2.5 feet by tuning the transmission power of each down to -5dBm.

We modified the Surge application, which uses a tree-based routing algorithm for a single network sink. Since our focus was on channel-oriented networking issues associated with interference resistance, we did not employ message acknowledgements or retransmissions. In addition to this basic communication model, we note that each sensor message contains a sequencing field (in the routing header) that can be used to estimate performance statistics, i.e. link quality [9]. Finally, we note that our packet size was 32 bytes, and that a node can buffer at most 24 packets (across all channels).

5.2 Building a Jamming-Resistant Network

After preparing the underlying sensor network testbed, we next implemented the channel surfing framework and the three strategies. In the implementation of these strategies, we faced several challenges. First, we modified the buffering mechanism to address the fact that buffered packets may need to be sent on different channels.

The second issue we addressed was related to the replacement policy of the mote neighbor table. Each Mica2 mote maintains a neighbor table recording the link quality between each node in its radio range (e.g. *radio neighbors*) and itself. The motes in our testbed only have 16 entries in their neighbor table. The default policy in TinyOS sorts the radio neighbors based on the link quality. This policy, however, must be modified to implement channel surfing strategies. To understand this, suppose node A's child, B, is jammed. A is supposed to find the link quality between B and itself has degraded, and then probe the next channel to search for B. However, since B is jammed, the estimated link quality between A and B may be so low that B is evicted from A's neighbor table. In this case, A loses awareness of the existence of B, and will not look for it on the other channel. In order to address this problem, we modified the replacement policy so that it always sorts a node's topological neighbors (i.e. the parent node and children nodes) ahead of non-topological radio neighbors.

The third issue is related to link quality estimation. Estimating the link quality involves comparing the number of packets a node receives with the number of packets it expects by using the sequence numbers of the packets. However, when we operate on multiple channels, this estimation mechanism may cause problems because a dual-mode node will have roughly a 50% link quality, and thus nodes on both channels will not choose the dual-mode nodes as their parents, hindering the realization of spectral multiplexing. We address this by assigning independent sequence numbers on different channels, so that the estimated link quality for a dual-mode node on both channels is sufficiently high.

5.3 Performance Metrics for Channel Surfing

In this study, we used the following performance metrics to evaluate the effectiveness of channel surfing strategies:

- *Network Recovery:* The main objective of channel surfing is to ensure network availability in the presence of jamming. Therefore, we first measure whether the channel surfing strategies can restore network performance (number of packets delivered to the sink) from jamming. Further, we also measure the latency required for these strategies to recover network performance.

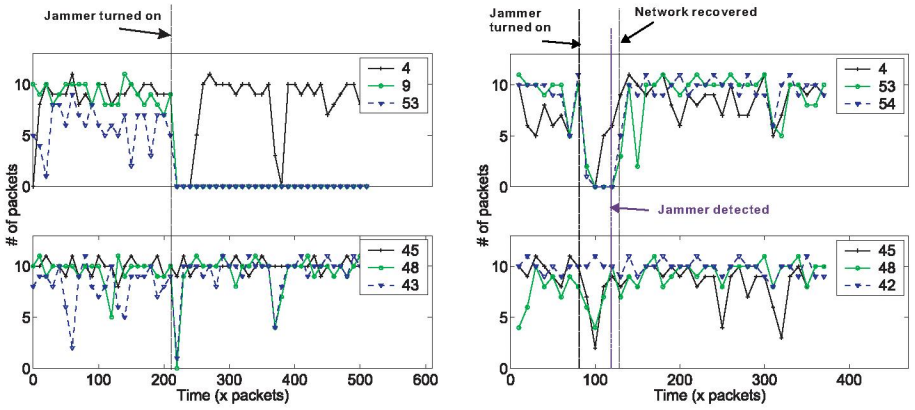


Figure 6: The number of packets delivered to the sink in every 10-packet window throughout the experiment for (a) first normal and then jammed network conditions, and (b) the coordinated chanel surfing strategy.

- *Protocol overhead:* Another class of metrics are related to the overhead introduced by the channel surfing strategies. One obvious concern is that nodes may unnecessarily switch channels. As a result, we have measured the number of channel switches every node experiences throughout the duration of an experiment.

6. EXPERIMENTAL RESULTS

6.1 The Impact of Jamming/Interference

We deployed the testbed as illustrated in Figure 5(a), and the resulting tree-shaped routing topology (Figure 5 (b)) was captured using the Surge Network Viewer. In the routing tree, the root node, node 0, corresponds to the network sink. We then conducted an experiment to study the network behavior under normal indoor conditions as well as the impact that a single jammer can have on the network. In this experiment, we first let the network run for more than 20 minutes prior to introducing a jammer. For our jammer, we used the same device as legitimate nodes, thus the jammer can only jam one channel at a time. In particular, we used the constant jammer of [2], which is a mote that bypasses the MAC-layer to continually transmit random bits into the network. Figure 5 (c) shows the location of the jammer, and the fact that the jammer destroyed the connections between several nodes and the sink. Specifically, in this example, nodes $\{9, 10, 33, 52, 53\}$ lost their connections because they were directly affected by the jammer, while nodes $\{1, 2, 3, 4\}$ lost their connections because their parents were jammed.

We present time series for the number of packets delivered to the sink in a window of 10 packet intervals (that is, during this window, each sensor node generates 10 packets) from six randomly chosen nodes in Figure 6(a). In these results, each node generated and sent packets at a rate of 1 packet every 5 seconds. Under a perfect network condition, we expect each node to deliver 10 packets in a 10-packet window, but even under normal network conditions prior to jamming, the traces exhibit short-term fluctuations. After introducing the jammer, nodes 9 and 53 were not able to deliver packets to the sink. As discussed earlier, that is because either these nodes were jammed, or all their possible parent nodes were jammed. As a result, the packet delivery ratio became 0. We note that although node 4 lost its former parent due to jamming, it later found a replacement parent via normal route discovery on channel 1 and hence its packet delivery became normal after a short interruption.

6.2 Coordinated Channel Switching Results

We conducted experiments to study the effectiveness of the coordinated channel switching method described in Section 4.1, in which the entire network changes its operating channel to a new channel. We observed that the coordinated channel switching method completely restores connectivity for every sensor to the network sink on the new channel as indicated by the packets-delivered time series shown in Figure 6(b). Here we selected a sampling of six nodes, and observed that regardless of a node's position, they can resume operations on the new channel quickly and almost at the same time. The switching latency is roughly the sum of jamming detection latency, probing time, and the broadcast latency. Specifically, the transition phase only took 46 packet intervals, and 39 out of the 46 intervals were used for the jammed nodes to detect they are jammed as well as for the boundary nodes to find out their children nodes are missing. We would like to emphasize that we purposefully let a node wait for a rather long time period (i.e. 39 packet intervals) before probing the next channel after it detects a poor link quality between itself and its children. *We take the viewpoint that sensor networks will experience a LOT more temporary topological changes than longer-term jamming/interference, and that we therefore must reduce the false positives of channel probing to achieve more stable network operations.* Also, we would like to point out that even with such a conservative approach, we could improve the recovery process by having jammed nodes buffer packets during the network transition periods.

We also measured the total number of channel switches for these six nodes versus time. As expected, we saw that, prior to introducing the jammer, no nodes switched channels because our algorithms were tuned to have very low false positives on determining whether to probe the next channel. As soon as the jammer started, since nodes 53 and 54 were directly affected, they switched channels, and stayed there afterwards, thus switching only once. Node 42, however, reported 3 channel switches because it was a boundary node that first switched to channel-2 probing for its child, then switched back to the original channel to broadcast the switch notice, and finally switched back to the new chan-

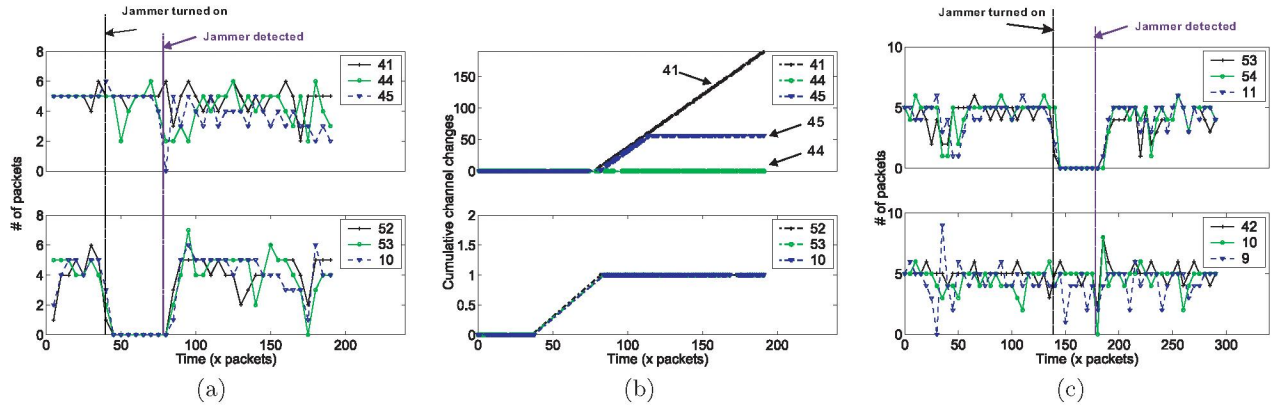


Figure 7: Statistics for the spectral multiplexing strategies: (a) packet delivery time series in a 5-packet window for the synchronous case, (b) total number of channel switches vs. time for the synchronous case, (c) packet delivery time series for asynchronous spectral multiplexing.

nel to resume network operations. Other boundary nodes exhibited similar behavior, while more distant nodes only switched once. Overall, the coordinated channel switch incurs very few channel switches. Finally, we note that we also conducted experiments with multiple simultaneous jammers in different positions, and observed that the coordinated method was able to repair the network in these cases with latencies on the order of 50 packet intervals.

6.3 Spectral Multiplexing Results

We now examine spectral multiplexing. In these methods, we only switch channels for the jammed nodes while a subset of nodes multiplex between the original and new channels. We note that it is unfair to compare coordinated channel switching with spectral multiplexing under the same network configuration because these two strategies are complimentary to each other, each designed to deal with different situations. Specifically, the coordinated strategy is suitable for cases where a large region of the network is jammed, while the multiplexing strategy is suitable for cases with much smaller jammed regions. Thus, for spectral multiplexing we block fewer nodes to have a smaller jammed region. One further difference between these two types of strategies is that spectral multiplexing typically requires more buffer space for storing packets during multiplexing. Given the limited buffer space on the motes, we chose to adopt a lower data rate (1 packet every 10 seconds) than the rate in earlier experiments.

6.3.1 Synchronous Spectral Multiplexing

In the synchronous spectral multiplexing experiment, we used the same general layout as shown in Figure 5(a). After the network had run for 40 packet intervals, the jamming/interference process began. The jammed region consisted of nodes 52, 53, and 10, and these three nodes promptly switched to channel 2. After the jammed nodes evaded to the new channel, their former parents detected their disappearance, and became the boundary nodes. Nodes 41 and 45 were such examples because they used to be the parents for nodes 10 and 52, respectively. The boundary nodes then announced themselves on the new channel, and waited to be selected as parents by the nodes on channel 2. In this experiment, all three jammed nodes chose node 41 as their parent. Thus, node 41 started working on two channels, while node 45 went back to work on channel 1. Overall, node 41 had four child nodes, three on channel 2 (nodes 52, 53, and 10), and one on channel 1 (node 44). In this experiment, the slot duration was 6.1 seconds, so that the number of packets buffered per channel was roughly 3 to 6 packets.

Figure 7(a) presents the time series of the number of packets delivered to the sink from six nodes in a 5-packet window. These six nodes include the three jammed nodes (nodes 52, 53, and 10), one boundary node (node 41), one potential boundary node (node 45), and one child of the boundary node that worked on channel 1 (node 44). The plot shows that the jammed nodes resumed their normal packet delivery performance. All other nodes were not affected much by the jammer. The interval during which the jammed nodes had disrupted services was roughly 48 packet intervals, which is similar to the recovery time in the coordinated channel surfing strategy. Again, during the total recovery latency of 50 packet intervals, the boundary nodes waited for around 39 packet intervals before switching to the new channel to search for their children. After the boundary nodes found their children on the new channel, it only took them 11 packet intervals to start working on dual channels.

We report the total number of channel switches for these six nodes in Figure 7(b). These numbers agree with the discussion above regarding how different nodes responded to the emulated jamming in the experiment (e.g. node 41 continually switches channels). Interestingly, as noted earlier, node 45 started out as a dual-mode node and it initially switched channels frequently. However, after a short period of time, it was not selected as a dual-mode parent for any nodes on channel 2. It then returned to channel 1 and no longer switched channels.

6.3.2 Asynchronous Spectral Multiplexing

Our asynchronous experiment used a similar setup as the synchronous multiplexing experiment. The jammed region consisted of nodes 53, 54, and 11. After the jammed nodes evaded to channel 2, their former parents, i.e. nodes 42 and 10, also switched to channel 2, and announced their willingness to work on channel 2. Node 42 was chosen to be the parent by all three jammed nodes, and started flipping between both channels in a round-robin manner, while node 10 continued on channel 1. From then on, node 42 had three children (the three jammed nodes on channel 2).

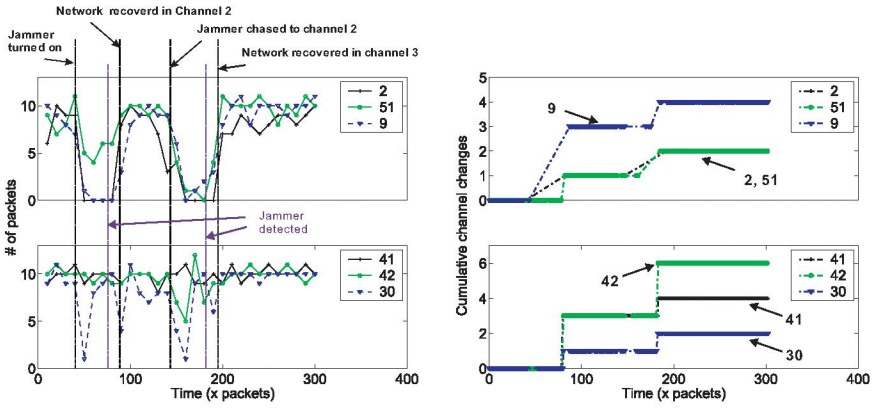


Figure 8: (a) Packet delivery time series for the coordinated channel switch strategy when the jammer follows the network's channel surfing. (b) Time series illustrating the amount of times a node has changed its channel during the network's operation.

Figure 7(c) presents the time series for the number of packets delivered to the sink from the above-mentioned nodes in a 5-packet window: the three jammed nodes (nodes 53, 54, and 11), one boundary node (node 42), and one potential boundary node (node 10). It is clear from this figure that the asynchronous multiplexing scheme can quickly recover the connections between the jammed nodes and the sink without affecting other nodes in the network. As in the case of synchronous multiplexing, this scheme also incurred roughly a 50-packet service disruption period for the jammed nodes. We also recorded the total number of channel switches for these six nodes and observed trends analogous to those reported for synchronous multiplexing.

6.4 Channel Surfing Discussion

Though the results presented above have shown that the three channel surfing strategies fare comparably, we emphasize that it is better to evaluate these strategies according to the network and interference scenarios for which they are most appropriate. Coordinated channel surfing, which requires all network nodes to switch their channel, is most suitable for cases where a large region is jammed, and the jamming occurs on a longer time scale (e.g. from a long-duration unintentional interference source). Spectrum multiplexing, however, is more effective for transient jamming where a few nodes are affected for a short duration. Here, we should determine whether to adopt synchronous or asynchronous spectral multiplexing based on the underlying traffic model. For instance, synchronous spectral multiplexing is more suitable for regular traffic patterns, while asynchronous spectral multiplexing can better cope with irregular (e.g. bursty) traffic.

6.5 Channel Following Jammers

We were also interested in more challenging interference scenarios, such as when the jammer follows the network as it channel surfs. We conducted experiments with this jammer model, and found that all the three proposed schemes could restore network connectivity in such cases. Due to space limits, we do not provide results for all three schemes here, but rather show the experimental results for the coordinated channel switch scheme.

The network setup was the same as in Figure 5(a). We started the network on channel 1, and then introduced the jammer approximately at the 40th packet interval, as depicted in Figure 8 (a). Shortly thereafter, the network adapted, with all nodes switching to the second channel after an overall latency of approximately 47 packet intervals. We assumed that the jammer took 50 packet intervals to find the new channel the network moved to, and thus the jammer switched to channel 2 at the 140th packet interval. The network again adapted to the interference, switching to the third channel after a total latency of roughly 51 packet intervals. Examining the packet delivery time series for node 51 illustrates an interesting phenomena regarding the effectiveness of a jammer: when the jammer was on channel 1 it was not entirely effective in disrupting the operation of node 51 while when the jammer was on channel 2, it was more effective at disrupting the communications from node 51. We believe this is due to the irregularity of the Mica2 radio. In addition to packet delivery traces, we recorded the amount of times different nodes switched channels, as shown in Figure 8 (b). This curve shows that the protocol does not require an excessive amount of channel switching, typically requiring either one or three channel change attempts prior to settling on the new channel. We draw the reader's attention to the channel change trace for node 41 and node 42. During the first channel change, these nodes switched back to the original channel to broadcast the change channel command, thus requiring a total of 3 channel changes. However, after the jammer follows them to the new channel, both the radio dynamics and the underlying network topology have changed, and in this case only node 42 was involved in switching back to channel 2 to announce the change channel command.

7. RELATED WORK

Coping with jamming and interference is usually a topic that is addressed through conventional PHY-layer communication techniques. In these systems, spreading techniques (e.g. frequency hopping) are commonly used to provide resilience to interference [4, 5]. Although such PHY-layer techniques can address the challenges of an RF interferer, they require more advanced transceivers and have not found widespread deployment in commercial sensor networks. We note that our frequency scheduling and synchronization meth-

ods are similar to those used in physical layer frequency hopping and TDMA [14], though our techniques operate on a much coarser time-scale.

The issue of detecting and mapping jamming for sensor networks was studied by Wood and Stankovic in [15]. The problem of jamming detection was further studied by Xu et al. in [2], where the authors presented several jamming models and explored the need for more advanced form of detection algorithms to identify jamming. Additional jamming strategies were studied by Law et al. [1], and the efficiency of these methods was quantified in terms of the amount of resources needed to conduct an attack. Further work on jamming has studied MAC-layer jamming attacks on reservation-based medium access control schems [16].

Countermeasures for coping with jammed regions in wireless networks has been studied in [6, 17–19]. In [17], the use of error correcting codes is proposed to cope with jamming. In [6], two countermeasures are presented for coping with jamming. The first method, channel surfing, serves as the motivation for this paper. The second method, spatial retreats, was studied in more detail in [18] and involves mobile nodes physically moving away from the interference to reestablish connections. In [19], wormhole-based antijamming techniques are studied analytically. The proposed approaches try to ensure the successful delivery of at least one alarm message for each event, instead of resuming continuous network connectivity. Our work builds on these efforts by providing system validation.

The use of multiple channels has been proposed as a means to enhance the throughput and performance of wireless mesh networks, e.g. [20–22], and cellular networks. In this area, the radio devices are resource-rich compared to sensors, and often have multiple-radio interfaces. The challenge here is assigning the channels and/or time slots across multiple nodes to avoid collisions and congestion. These works are primarily intended to enhance performance in normal conditions, and do not attempt to cope with unexpected jamming/interference, as is described in this paper. Generally, channel allocation for these scenarios is achieved with the aid of a centralized entity or via a common control channel. In our work, we have not resorted to centralized entities or control channels in order to achieve jamming resistance.

8. CONCLUDING REMARKS

Many commercial wireless sensor networks are susceptible to radio interference/jamming. To ensure the availability of sensor communications, interference defense mechanisms must be developed that are distributed, easy to scale, and have low false positives. We have tackled this challenge in this paper by presenting a family of channel surfing strategies that can restore connectivity in the presence of radio interference. We presented two families of channel surfing strategies: the first, coordinated channel switching, involves the entire sensor network changing its operating frequency; the second family, which we refer to as spectral multiplexing, changes the operating frequency in a neighborhood local to the interference, with boundary nodes acting as a radio-bridge across different channels. We implemented our strategies on a testbed of Mica2 motes, and have reported their performance for several interference scenarios. We found that our broadcast-assist strategy, as well as our multiplexing schemes, can effectively repair the network with short latency. Our study serves as an initial systems effort towards building interference-resistant sensor networks and future investigations will involve examining more advanced jamming scenarios.

9. REFERENCES

[1] Y. Law, P. Hartel, J. den Hartog, and P. Havinga, "Link-layer jamming attacks on S-MAC," in *Proceedings of the 2nd European Workshop on Wireless Sensor Networks (EWSN 2005)*, 2005, pp. 217 – 225.

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

[3] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks," in *Proceedings of the Usenix Symposium on Operating Systems Design and Implementation*, 2002.

[4] J. G. Proakis, *Digital Communications*, McGraw-Hill, 4th edition, 2000.

[5] C. Schleher, *Electronic Warfare in the Information Age*, MArtech House, 1999.

[6] W. Xu, T. Wood, W. Trappe, and Y. Zhang, "Channel surfing and spatial retreats: defenses against wireless denial of service," in *Proceedings of the 2004 ACM workshop on Wireless security*, 2004, pp. 80 – 89.

[7] A. Wood and J. Stankovic, "Denial of service in sensor networks," *IEEE Computer*, vol. 35, no. 10, pp. 54–62, October 2002.

[8] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003, pp. 1–13.

[9] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003, pp. 14–27.

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

[11] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *Proceedings of the IEEE INFOCOM*, 2002, vol. 3, pp. 1567– 1576.

[12] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing-sync protocol for sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003, pp. 138–149.

[13] "Tinyos homepage," http://webs.cs.berkeley.edu/tos/.

[14] S. S. Rappaport and D. M. Grieco, "Spread-spectrum signal acquisition - Methods and technology," *IEEE Communications Magazine*, vol. 22, pp. 6–21, June 1984.

[15] A. Wood, J. Stankovic, and S. Son, "JAM: A jammed-area mapping service for sensor networks," in *24th IEEE Real-Time Systems Symposium*, 2003, pp. 286 – 297.

[16] A. Rajeswaran and R. Negi, "Dos analysis of reservation based mac protocols," in *Proceedings of the IEEE International Conference on Communications*, 2005.

[17] G. Noubir and G. Lin, "Low-power DoS attacks in data wireless lans and countermeasures," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, no. 3, pp. 29–30, 2003.

[18] K. Ma, Y. Zhang, and W. Trappe, "Mobile network management and robust spatial retreats via network dynamics," in *Proceedings of the The 1st International Workshop on Resource Provisioning and Management in Sensor Networks (RPMSN05)*, 2005.

[19] M. Cagalj, S. Capkun, and J.P. Hubaux, "Wormhole-Based Anti-Jamming Techniques in Sensor Networks," *to appear in IEEE Transactions on Mobile Computing*, January 2007.

[20] A. Raniwala, K. Gopalan, and T. Chiueh, "Centralized Channel Assignment and Routing Algorithms for Multi-Channel Wireless Mesh Networks," *ACM Mobile Computing and Communications Review*, vol. 8, no. 2, pp. 50–65, 2004.

[21] J. So and N. Vaidya, "Multi-channel MAC for ad hoc network: Handling multi-channel hidden terminals using a single transceiver," in *Proceedings of ACM MobiHoc*, 2003, pp. 222 – 233.

[22] R.Garces and J. G. L. Aceves, "Collision avoidance and resolution multiple access for multi-channel wireless networks," in *Proceedings of IEEE INFOCOM*, 2000, pp. 595–602.