Poster Abstract: Channel Surfing: Defending Wireless Sensor Networks from Jamming and Interference

Wenyuan Xu
WINLAB
Rutgers University
Piscataway, NJ 08854
wenyuan@winlab.rutgers.edu

Wade Trappe
WINLAB
Rutgers University
Piscataway, NJ 08854
trappe@winlab.rutgers.edu

Yanyong Zhang
WINLAB
Rutgers University
Piscataway, NJ 08854
yyzhang@winlab.rutgers.edu

Categories and Subject Descriptors

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

General Terms

Security

Keywords

Jamming, Radio Interference, Channel Surfing

1 Introduction

Whether intentional or not, interference and jamming will be a serious threat to the reliable communication of sensor messages [3]. The traditional approach to coping with radio interference is to employ more sophisticated physical-layer technologies (such as spread spectrum). Such methods, however, imply more expensive transceivers and, with the exception of some military systems, most commodity sensor and wireless networks do not employ sufficiently strong spreading techniques to survive jamming or to achieve multiple access. Instead, systems like the Berkeley Mica2, the Zigbee and even 802.11 are based upon a carrier-sensing approach to multiple access. Because of their use of carrier sensing for medium access control, these systems are particularly susceptible to radio interference or jamming. Recent studies [2], have revealed the relative ease with which jamming can be conducted on such sensor networks. In this paper, we examine the ability of a sensor network to cope with radio interference or jamming. We propose the use of *channel surfing*, where jammed nodes switch to a different channel and boundary nodes multiplex between two channels. In order to validate these strategies, we have implemented the proposed methods on a 30-node Mica2 mote sensor network testbed.

2 Channel Surfing Strategies

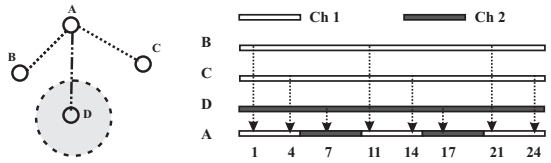
In channel surfing, those nodes that detect themselves as *jammed* nodes [1, 2] should immediately switch to another orthogonal channel. Their neighbors, which we refer to as *boundary* nodes, will discover the disappearance of their jammed neighbor nodes and temporally switch to the new channel to search 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. Although it is possible to devise schemes where the entire network changes its channel starting from the boundary nodes, such a strategy has drawbacks (e.g. latency) and consequently it is desirable to have the rest of the network stay on the original channel, and have the boundary nodes multiplex between the old channel and the new channel. We call this scheme Spectral Multiplexing. The primary challenge with this scheme is to carefully decide when the boundary nodes should stay on which channel, and for how long, so that they can minimize the frequency mismatch between the sender and the receiver.

2.1 Synchronous Spectral Multiplexing

In synchronous spectral multiplexing, the entire network is governed by one global clock. The global time axis is divided into slots.Each slot is assigned to a single channel, and during that time slot, the network 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 dual-mode boundary nodes. By following this global schedule, we can avoid frequency mismatch between a pair of communicators.

The most challenging issue with this scheme is how to synchronize the schedule of every node. Having a global synchronized clock, is not only inefficient, but also unnecessary. Instead, since communication takes place locally amongst neighboring nodes, we can focus on achieving a fine synchrony within any local region. In our implementation, nodes use timers to demarcate slots, and to synchronize the timers on different nodes, *SYNC* packets are sent periodically at an interval much larger than the slot duration. In a tree-based routing structure, the network-wide synchronization process works as follows. In the first round the root first sends out *SYNC* packets to its children, whose depth is 1. Similarly, in the $(i+1)^{th}$ stage, the nodes with depth i send *SYNC* packets to their children. The synchronization process can become tricky when the parent node is on channel 1 and tries to send a *SYNC* packet to a node on channel 2. We have addressed this complication in two ways: (1) we have the

SenSys'06, November 1–3, 2006, Boulder, Colorado, USA.
ACM 1-59593-343-3/06/0011



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

Figure 1. Illustration of the round-robin asynchronous spectral multiplexing algorithm.

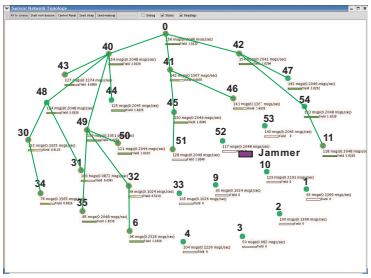


Figure 2. The network topology shortly after the jammer is introduced.

SYNC packet specify the channel associated with the current slot, and (2) we have every node send *SYNC* packets in rapid succession across both channels.

2.2 Asynchronous Multiplexing

In the asynchronous multiplexing algorithm, a node is only aware of its neighbors' channel information. The simplest spectral scheduling method is to have a boundary node flip its radio frequency between two channels in a round robin fashion. Figure 1 illustrates such a multiplexing schedule, adopted by *A* who is to receive packets from *B* and *C* on channel 1, and *D* on channel 2. The arrows in the figure illustrate message transmissions. Using the illustrated schedule, *A* can receive every packet from its neighbors.

In order to coordinate the schedules of a boundary node and its children, we let the boundary node notify its children just after it switches to a new channel and before it leaves a channel. Another challenge involves determining how long a boundary node should stay on each channel. While the boundary node should stay on each channel long enough to offset the switching overhead, the stay time is limited by the amount of buffer space available on its children who are working on the other channel. Also, it is partially limited by the buffer space available on the boundary node itself: a boundary node on channel 2 cannot immediately forward packets it receives because its parent resides on channel 1.

Compared to synchronous multiplexing, asynchronous multiplexing does not maintain a global schedule, and thus in- curs less synchronization overhead. The advantage of asynchronous multiplexing, however, is more pronounced when the jammed region is small and regular.

3 Experimental Results

We have built our sensor network testbed using 30 Mica2 sensor motes. 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. Given the limited buffer space on the motes, we chose to adopt a low data rate (1 packet

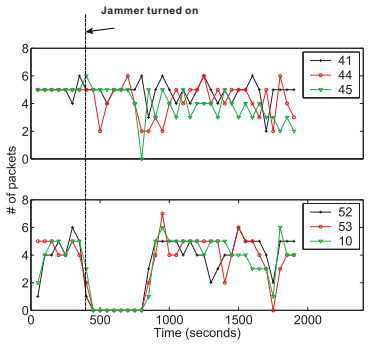


Figure 3. Packet delivery time series for synchronous spectral multiplexing.

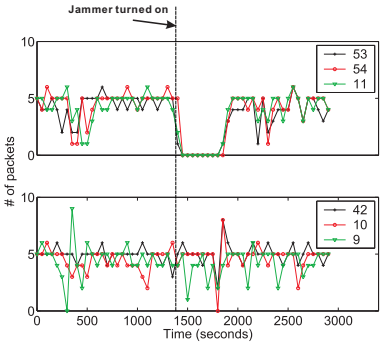


Figure 4. Packet delivery time series for asynchronous spectral multiplexing.

every 10 seconds) in our experiments. In a densely-deployed network like the one we had (nodes are separated from each other by 2.5ft), it is rather challenging to control the jammed region by using a real jammer. Thus, since we wanted to have a small jammed region, we emulated the effect of a jammer by staging a subset of nodes to believe they were jammed (rather than jam the nodes with with a real jammer). Figure 2 presents the topology shortly after jamming occurs.

Figure 3 presents the time series of the number of packets delivered to the sink from six nodes in a 50-second window under synchronous spectral multiplexing. Among these six nodes, only the bottom three were jammed, and the results show that these jammed nodes first suffered poor packet deliver ratio due to jamming, but then could resume their normal deliveries after 480.1 seconds (during which approximately 48 packets were lost).

The packet delivery time series for asynchronous spectral multiplexing, presented in Figure 4, are similar as the results of synchronous spectral multiplexing. More extensive experimental scenarios have been examined and are not presented due to space limitations.

4 References

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

[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*, pages 46–57, 2005.

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