MAC Stability in Sensor Networks at High Network Densities

ISI-TR-628, January 24, 2007

Tyler McHenry John Heidemann

Information Sciences Institute, University of Southern California
4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292
{smchenry,johnh}@isi.edu

Abstract

Although MAC protocols have been the subject of extensive study, there has been little study of MAC operation as network density (number of neighbors per node) increases. Although network densities are often low (2-12 neighbors), density can rise in special situations (such as hundreds of people attending a conference in the same room) and new deployments (such as dense instrumentation of a structure with a sensor network). In anticipation of these applications, this paper studies the stability of S-MAC as network density increases to densities of 50 to 150 neighbors, well beyond its current design parameters. We present a mathematical model of expected behavior, then use experiments to show the importance of accounting for clock offset. Although offset cannot easily be modeled, we show that a simulation closely matches the experimental data. Finally, we describe how an offset-aware MAC can correct for hardware variation to allow operations at twice the density of current S-MAC. Although the details are specific to S-MAC, the results apply more generally.

1 Introduction

Wireless MAC protocols are designed to promote efficient and often fair access to the network. In sensor networks, MAC protocols are often optimized to reduce energy consumption. However, most wireless networks today are relatively *low density*—each node has a relatively few neighbors, or each base station has relatively few clients.

Low-density networks are common in sensor network deployments today, where typical network deployments are sparse (fewer than 6–12 neighbors per node), or even engineered to run at very sparse but connected densities (2–6 neighbors). Sparse deployments maximize spatial reuse, and a class of power-adaptive protocols adjust node transmit power to create topologies with 12 or fewer neighbors per node [8, 5, 2].

While sparse networks are common today, we argue that there will also be much denser networks in the near future. Falling costs of sensornet hardware will make dense deployments feasible, demand for better coverage and reliability make dense deployments desirable, and limited ability to adjust transmit power makes density unavoidable. Already, high densities arise in special circumstances, such as when an 802.11 network moves from a quiet home to a bustling conference room with hundreds of chattering laptops. In sensor networks, sensor placement is often driven by the need to sense certain phenomena; applications such as structural health monitoring force increasingly dense deployments [12]. Visionaries also anticipate "smart paint" with thousands of wireless computing elements per square meter [1]. Wireless MAC protocols typically have not been studied at high network densities.

In anticipation of these applications, this paper studies the stability of S-MAC [14] as network density increases to 50 to 150 neighbors, well beyond its current design parameters. While in the abstract, MAC protocols can operate at any density, in practice, an implementation includes many internal parameters that are selected with some density in mind. Our goal is to understand how a deployed MAC protocol must adapt to scale to both low and high densities.

Nodes must send control messages such as RTS/CTS and, in S-MAC, synchronization packets. As density rises, these messages can encounter collisions and loss. Approaches such as randomization and retry may work well at low densities, but they eventually break down. While congestion control at higher network layers can successfully increase the overall throughput of a network, it does not resolve the stability of MAC-layer control messages.

The main contribution of this paper is to investigate MAC reliability at high densities. We present a mathematical model of expected behavior. We then use experiments in a scaled network to show that hardware effects such as clock offset actually invalidate simple models. Although offset cannot easily be modeled, we show that an simulation closely matches the experimental data. Finally, we describe how an offset-aware MAC can correct for hardware variation to allow operations at up to twice the density of current S-MAC. Although just a beginning, we believe this paper opens the door to future exploration of high-density sensor networks.

2 Problems from Density

Several problems arise when network density grows, even if application traffic remains constant. First, the pre-allocated, fixed-size neighbor table must be large enough to include all neighbors. Second, the soft-state approach to maintaining this table must operate correctly. Third, channel contention must scale as the number of active nodes grows. In this paper we assume that the neighbor table is large enough and we focus on the cost of its maintenance.

In Section 3.1 we will describe the soft-state protocol that S-MAC uses to maintain the neighbor table. A node appears in the tables of its neighbors provided that it successfully sends synchronization packets at a rate meeting or exceeding a specified minimum frequency. We call nodes that do not meet this criteria *timed out*, and if there is at least one such node, the network is said to be *desynchronized*. By definition, a network that is desynchronized contains at least one node A which is not considered a neighbor by at least one other node B, even though they are within radio range.

The cost of a desynchronized network is longer routing paths. Given the high density, it is likely that all nodes are still reachable, but nodes that are timed out will require two or more hops to exchange data through the MAC layer, even though they are physically within radio range. Since these extra hops waste power needlessly, we consider a desynchronized network undesirable.

To quantify network stability we define three values of interest. First is the network size at which the expected or mean number of timed out nodes is greater than one-half. We define this size as the *instability point*, because at this point the network is more likely than not to be desynchronized. The next is the network size at which the expected or mean number of timed out nodes is greater than one. This point is called the *break point* because here the network is perpetually desynchronized (broken) since at least one node is always unreachable for application-layer packets.

Finally, there is the network size at which the expected or mean number of timed out nodes is $n - 1$. This point is called the *total failure point* because here each node is not aware that it has any active neighbors, and may as well not be in a network at all. While this point is not investigated experimentally in this paper, it is important to consider, since this is the point at which every packet is expected to be undeliverable regardless of how many hops are allowed.

3 S-MAC Background

In this paper, we use the S-MAC protocol [14] as our example media access layer. We chose it because

it is fairly simple, a full software implementation is freely available, and because it is representative of contention-based sensor networking MACs that strive to conserve energy.

There are a number of aspects of S-MAC that are sensitive to density. In this paper we focus on the frequency of synchronization messages, the keep-alive message that is vital to the operation of the protocol.

3.1 Protocol Elements

S-MAC aims to allow most nodes to be in a low-power sleeping state most of the time. Nodes periodically wake up and exchange information on a particular schedule. Nodes will wake up at a pre-arranged time, at which point they will listen for and transmit synchronization packets. These packets synchronize schedule timing and confirm neighbor presence. Then, if necessary, the nodes exchange RTS and CTS packets to reserve the channel for data transfer. After this exchange occurs, nodes not participating in a unicast transfer return to sleep, freeing the medium for the data transfer.

This paper will follow the terminology used in prior S-MAC papers. A stream of communication over the S-MAC protocol can be decomposed into *frames* and *periods*. A frame is the basic unit of S-MAC communication; in basic usage it provides the opportunity to send one data packet plus synchronization packets. S-MAC synchronizes all nodes so that all nodes know when frames begin and end. Nodes periodically exchange synchronization packets to maintain their synchronization. A period is the default interval between a nodes transmission of a synchronization packet.

Each frame is divided into three segments: the synchronization segment, the contention segment, and the data transmission segment. The synchronization segment is composed of some number of *slots*, which each represent an opportunity for one node to capture the channel for sending a synchronization packet. Several slots are provided to allow randomization to reduce the likelihood of collision.

The contention segment allows a node to acquire the channel through an RTS/CTS exchange. In the data segment, the source and destination of a packet remain awake while all others sleep. (For broadcast traffic, all remain awake.) At low duty cycles, data exchange may be followed by a quiet period. With adaptive listen [14], or if a transmission is lost and is not acknowledged by the receiver, additional contention and transmission periods may follow in one frame and even cross into subsequent frames. We do not consider that level of detail in this paper.

Each node in S-MAC keeps a table of its immediate neighbors. The main purpose of this table is to track active schedules, since multiple independent schedules can arise. When multiple schedules are present, this table is used to determine which schedule is best for a given destination.

Since nodes may fail, turn off, or move, S-MAC tracks nodes and requires all nodes to indicate their schedules periodically with synchronization packets. A node that does not successfully transmit synchronization packets frequently enough is said to *time out* and is considered to no longer be a member of the network until it sends another synchronization packet.

The experiments in this paper are concerned with measuring, modeling, and predicting the occurrences of spurious timeouts in a fully-connected S-MAC network as a function of network size. This data will give a quantitative view of the robustness of the protocol as a function of density.

3.2 Tuning S-MAC

S-MAC has a number of parameters, including the number of synchronization header slots per frame, frames per period, and periods per timeout. In this paper we use the variables k, s, and t to represent these values, respectively. We also use n to refer to the number of nodes in the network, and we assume that all nodes can hear one another (the worst case for high density).

Synchronization packets are actually larger than a single synchronization slot. Thus, slots serve to reduce the probability collision (since all nodes carrier sense and require a clear channel before transmitting), but we assume each synchronization packet consumes r slots.

In S-MAC models and simulations, we use the indices i to indicate a particular frame and j to indicate

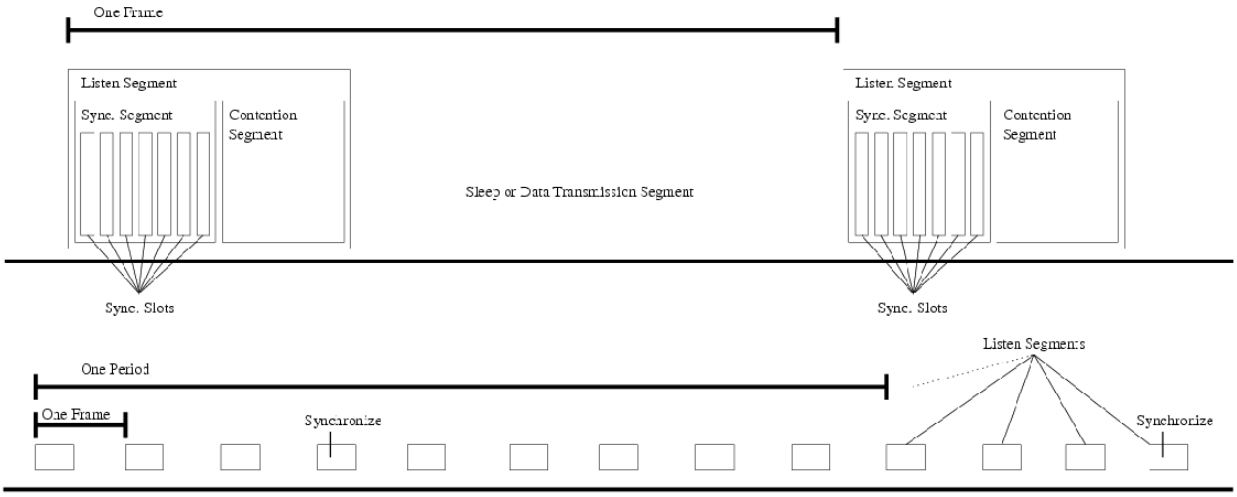

Figure 1: Conceptual diagram of S-MAC slots, frames, and periods.

a particular slot.

The reference implementation of S-MAC uses 15 slots per frame ($k = 15$ slots) and 12 periods per timeout ($t = 12$ periods). The intention is that there be 10 frames per period, but due to implementation details, for S-MAC with no customizations, there are in practice only 9 ($s = 9$ frames), which is what is used in this paper. Note that st gives a measure of frames per timeout.

The length of a synchronization packet (r) has been measured empirically to be approximately 13.5 milliseconds, or approximately 6.75 slots. With carrier sense, this consumes seven slots, so we set $r = 7$.

3.3 Neighbor Maintenance

S-MAC uses synchronization packets to maintain neighbor tables and schedule tables. They are essentially a "keep-alive" message from one node to its neighbors, informing them that it remains an active part of the network. Ignoring the rest of the operation of S-MAC, the mechanism for keeping the nodes synchronized with one another is as follows:

- Each period, a node will select one of the k synchronization slots uniformly and independently at random and carrier sense from the beginning of the frame until the beginning of that slot.
- If the node detects a free channel at the beginning of the slot, it will begin transmitting its synchronization packet (which will then occupy the medium for r slots).

In S-MAC, broadcast packets, including synchronization packets, are not controlled with RTS/CTS or any other method of reserving the channel, so they are subject to collisions. This paper will assume that all packets properly transmitted are properly received. From our experience, the primary cause of lost packets is due to collisions, not other kinds of environmental noise. We consider collisions next.

3.4 Collisions and Preemption

Three things can cause a node to be unsuccessful in its attempt to send its synchronization message. Two nodes may randomly pick the same slot and *collide*, or the node may detect that the channel is busy and abort its transmission, being *preempted*. Following a preemption, a node skips its transmission in the current frame and attempts again in the following frame with a new, randomly chosen slot. There are two

forms of preemption: A *normal preemption* occurs when the channel is occupied because another node that began transmitting its synchronization packet on an earlier slot has not yet finished. The more interesting situation is a *offset-related preemption*.

Offset-related preemption occurs when the channel is occupied because another node which is attempting to transmit on the *same* slot began transmitting early relative to the node being preempted. It may be because the preempted node's clock is offset late, or because the preempting node's clock is offset early, or both. Regardless, there must be a significant enough discrepancy between their clocks that the preempting node begins its transmission before the preempted node finishes carrier sensing. Otherwise, the preempted node would not have detected a busy channel, and would have assumed that the channel was free to acquire.

If the clocks for two nodes attempting to transmit on the same slot are sufficiently synchronized that each stops carrier sensing before the other begins to transmit, each will assume a free channel and will assume that it has acquired the channel. Then, they will transmit their synchronization packets almost simultaneously, and the packets will collide.

Collisions are much worse than preemption because the radios in question are simplex. A node cannot detect a collision of its own packet, and so must always assume that broadcast packets (which receive no ACK) are successful. Therefore, instead of retrying the synchronization on the very next frame, all nodes involved in the collision will instead wait an entire period before attempting another synchronization, as if they had succeeded. Fortunately for the protocol, the window of synchronization tolerance between two nodes for which a collision can occur is quite small compared to the observed clock offsets. Thus such collisions are, in practice, relatively rare.

4 An Idealized Model of S-MAC

Our original assumption was that the primary barrier to high-density operation would be collisions of synchronization packets. We therefore designed an idealized model to predict S-MAC stability.

4.1 The Idealized Model

The idealized model makes the following major assumptions:

1. Initially, the first synchronization attempts of all nodes are uniformly distributed over the frames in the first period.
2. There is no clock drift or offset; all clocks are perfectly synchronized.
3. There is no delay between carrier sensing and transmitting; a free channel after carrier sense always means either a free channel for transmission or an impending collision.

The idealized model ignores all preemptions resulting from non-ideal clocks; experimental results in Section 5 will show that this factor cannot be dismissed. The idealized model further assumes that it is always possible to transmit $\lceil \frac{k}{r} \rceil$ complete synchronization packets in each frame, which is a best-case scenario. Best-case analysis is appropriate for the idealized model since the model is ultimately pessimistic. Analyzing the best case demonstrates that the pessimism is inherent to the model and not due to overly stringent assumptions.

4.2 Analysis of Idealized Model

Since the idealized model assumes full synchronization, there are no offset-related preemptions, thus guaranteeing that there will be at most $\lceil \frac{n}{s} \rceil$ nodes attempting to synchronize in each frame. Each of these nodes will be competing for one of the $\lceil \frac{k}{r} \rceil$ available synchronization opportunities. The chance that a given node will succeed in synchronizing on that frame is the chance that the slot it chooses is unique among nodes attempting to synchronize on that frame. Since the choice of slot is uniformly at random, this is at least

$$\left(\frac{\lceil \frac{k}{r} \rceil - 1}{\lceil \frac{k}{r} \rceil} \right)^{\lceil \frac{n}{s} \rceil - 1} \tag{1}$$

Then, the probability that a given node fails to synchronize for t consecutive periods is simply

$$\left(1 - \left(\frac{\lceil \frac{k}{r} \rceil - 1}{\lceil \frac{k}{r} \rceil}\right)^{\lceil \frac{n}{s} \rceil - 1}\right)^t \quad (2)$$

And finally, since this probability is the same for every node, the expected number of nodes that are timed out on any given frame is the product of the number of nodes with the probability that any given node is timed out

$$n\left(1 - \left(\frac{\lceil \frac{k}{r} \rceil - 1}{\lceil \frac{k}{r} \rceil}\right)^{\lceil \frac{n}{s} \rceil - 1}\right)^t \quad (3)$$

This equation provides a closed-form evaluation of the expected number of unsynchronized nodes as a function of network size. The predictions of this model are plotted against experimental data in figure 2.

5 Experimental Observations

To verify our idealized model, we tested instability and break points of a range of moderate-sized networks. The key result of these experiments is to show the influence of systematic clock offset on network stability, as we explain below.

5.1 Scaling the Protocol

Analysis predicted a network break point of 37 nodes. Since we did not have enough nodes to test this prediction to the break point and beyond, we *scale* the S-MAC parameters down to cause instability with smaller numbers of nodes. We can then extrapolate from these results to a larger network with the default parameters.

The variables a_k, a_s, and a_t represent scale factors for the parameters k, s, and t respectively. Let $\hat{k}$, $\hat{s}$, and $\hat{t}$ be fixed at the default values, and to scale by the given factors we will let $k = \frac{\hat{k}}{a_k}$, $s = \frac{\hat{s}}{a_s}$, and $t = \frac{\hat{t}}{a_t}$.

In this particular experiment we chose to scale s and t by a factor of 3 but to leave k at its default value. So here, $a_s = a_t = 3$, and $a_k = 1$. It is important to note that the scale factors are used only to compensate for the lack of sufficient hardware to conduct a full-scale experiment. We do not claim that the scaling is necessarily linear (although it appears to be nearly so), but linear scaling is not required since we avoid comparing scaled data to unscaled projections.

5.2 Experimental Methodology

For this experiment, the nodes ran a no-op application that did not transmit any data of its own, and each mote ran S-MAC as its MAC layer protocol. We configured S-MAC to use a global schedule (initiated by node number 1) and to refuse to change schedules for any reason. This modification forced the collection of proximate nodes to maintain, or attempt to maintain, a single-schedule clique. Without these configurations, at densities beyond the breakpoint the nodes would likely have split into two or more parallel networks with different schedules. Since such a configuration is very wasteful of energy and precludes the study of grossly overloaded single-schedule networks, we explicitly prevented its formation.

We arranged 37 Mica2Dot motes on a table in a grid pattern of five nodes per row by eight rows (with the final row incomplete). Each node had approximately an inch and a half of space between itself and the next node both horizontally and vertically.

The data point for an n node network represents the data gathered by booting nodes 1 through n, allowing time for initial synchronization either by waiting 30 seconds or by waiting for each node to send at least one synchronization packet (whichever was longer), and then recording the complete activity of the network for four minutes. Given the scaling parameters used, four minutes of recording is equivalent to approximately 64 periods, or 192 frames.

A Mica2 mote situated in the incomplete area of the final row recorded network activity. This mote was running a snooper application (not using S-MAC) which listened constantly and echoed every received packet over a serial backchannel to a work-

station running a packet collection application. The packets were then analyzed on the workstation to determine which nodes were considered timed out at any given point in the experiment.

We recorded data points for network sizes from 6 to 20 nodes. After a size of 20, we stopped recording because the expected number of timeouts per frame reached 8—significantly past the break point. We repeated the above experiment three times.

5.3 Evaluation and Comparison to Model

As predicted by the idealized model, the mean number of nodes that were timed out during any given frame experienced a decidedly exponential increase. The exponential rate of increase reflects the direct relationship between the choosing of random slots and the well-known birthday problem [10].

The idealized results are in fact quite accurate at lower network sizes, but increasingly pessimistic as the network gets larger. The model correctly predicts the break point of the scaled experiment at a network size of about 9 nodes, but immediately afterwords the slope of the idealized model prediction becomes noticeably steeper than the experimental results. A direct comparison of the values is shown in figure 2.

The stepped pattern of the idealized results is interesting. This pattern is a result of the ceiling function in the inner exponent of (3). Because of this, the idealized model makes relatively static predictions for groups of three adjacent sizes with large jumps in between. The accuracy of the idealized model at low sizes is in large part due to the lack of opportunity for the idealized model to have made many of these large jumps at that point. This will be especially true for unscaled networks.

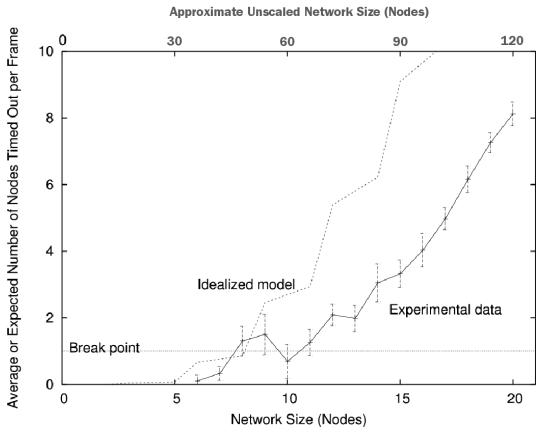

Figure 2: Expected and average timeouts per frame versus network size for a scaled experiment (with confidence intervals) and the scaled idealized model for 0 to 20 nodes.

6 The Impact of Hardware Effects on the Accuracy of Models

Our experimental results, however, match poorly at denser networks—our actual network is much *more* stable at high densities than we expected. We identified the inconsistency as a simplifying assumption in the model: we assumed clock synchronization was perfect. We next demonstrate that this assumption does not hold, and that clocks are actually systematically offset from each other. We show how this affects synchronization, and describe simulation results that better reflect reality.

6.1 Measuring Clock Offset

Ideally, all nodes are perfectly synchronized and so each begins its carrier sense at the same instant. With this assumption, the primary cause of instability is synchronization packet collisions due to selection of the same slot. In practice, we will show that accidental clock offset can be on the order of several slot lengths.

To measure the variability of clock timings, we selected eight nodes and designated one as a time reference. All nodes ran a no-op application on top of S-MAC (*i.e.* no data was being transmitted between nodes). The reference node booted once and continued to run. Then, the other seven nodes were one by one booted in close proximity to the reference node,

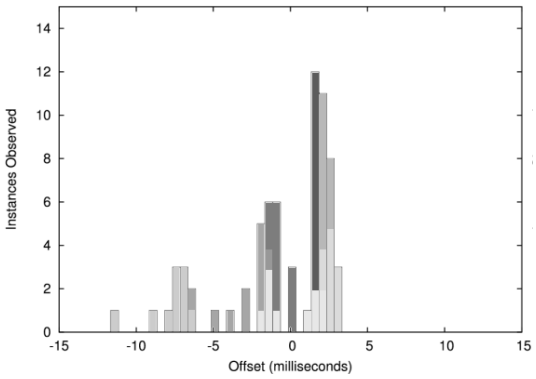

Figure 3: Histogram of measured clock offset distribution relative to stable reference node. Shading shows data gathered from individual nodes; notice the clustering.

allowed one minute to synchronize, and then the discrepancy between the two clocks was measured precisely ten times with an oscilloscope.

The left side of figure 3 shows the results of this experiment. While many nodes are accurate within a few milliseconds, we see that some nodes are more than 5 ms early. Recall that slot time is 2 ms, this means nodes are drifting by several slots. Also note that these measurements are all relative, since there is no master clock determining when each slot begins and ends. Rather, an arbitrary node among those measured was chosen as a reference point for the data.

In addition, we determined that clock offset is *systematic* based on the hardware. The right side of figure 3 shows the data from two nodes (5 and 7) after several cold reboots. Compared to the reference, we can see that node 5 is consistently early, while node 7 is always 2 ms late. In other words, some nodes seem to have clocks that are consistently early, and other seem to have clocks that are consistently late.

6.2 Effects of Systematic Clock Offset

Systematic clock offset has a significant impact on the stability of S-MAC. Some nodes become synchronization-greedy, since they consistently fire early and seize the channel, while others that are late become synchronization-deprived. When network density grows so that there is contention for nearly every slot, nodes with early clocks will "win" their slot regularly by beginning to transmit while other nodes that have selected the same slot are still carrier sensing. In contrast, late nodes will always yield the channel (losing carrier sense) and be unable to send their synchronization message even after repeated attempts.

Recall that the primary metric in the experiment described in section 5 is the mean number of nodes that are timed out in any given frame. The existence of greedy nodes and deprived nodes increases the mean number of timed out nodes beyond what an idealized hardware model would project. This desparity exists because deprived nodes, once timed out, are likely to continue to remain timed out for many more periods than would be anticipated were all nodes equally likely to "win" any given slot. This bias may make the likelihood of a network being desynchronized relatively high even if the likelihood of any one randomly selected node being timed out is relatively low.

The unfairness caused by this systematic bias is analgous to the network capture effect observed in Ethernet and other CSMA/CD networks [7]. In the case of S-MAC, a group of nodes, rather than a single node, are able to collectively capture the channel and temporarily starve the remaining nodes. Here, hardware differences, rather than the backoff algorithm, is the source of the unfairness, and the problem is complicated by the lack of collision detection in wireless networks.

6.3 Simulation Results

To correct our model, we must account for hardware clock variability. Since it becomes mathematically challenging to capture clock offset in analysis, we developed a custom simulator to confirm our hypothesis[1].

The simulator is a special-purpose design meant only to simulate the synchronization mechanism in S-

[1]Source code to the simulator and a written description of its operation are available at `http://www.isi.edu/~ilense/software/smac_sync_sim/`

MAC; it does not simulate the complete protocol, nor provide even a general, discrete-event simulation. It makes the same assumption as the idealized model regarding the first synchronization attempts being uniformly distributed. However, instead of assuming no offsets and no delay in the transition between carrier sensing and transmitting, the simulator approximates reality as follows:

1. A fundamental clock offset is assigned to each node at initialization; these offsets are selected at random from a normal distribution with a mean of zero and a standard deviation of 3.62 ms (the empirical mean).

2. An additional minor variation in offset is selected for each node every time it attempts to synchronize; these variations are selected from a normal distribution with a much smaller standard deviation half that of the fundamental offset, but also with a mean of zero.

3. The time by which carrier sense begin preceeds the beginning of the first slot is fixed at the empirical mean value of 21.3 ms.

4. The delay between carrier sense end and transmission is fixed at the empirical mean value of 0.68 ms.

6.4 Comparison of Simulation with Experimental Results

Figure 4 shows a comparison of the experimental and simulated results. The simulation results are slightly optimistic, predicting a slightly more stable network than experiment confirms. For the scaled network of the experiment, the simulation predicts the break point to be approximately 13 nodes. The experimental data shows a breakpoint at approximately 10 nodes. The difference is likely due to the inadequacy of random offset assignments for modeling the actual hardware effects. Importantly, however, the shape of the simulation results match the shape of the experimental data better than the idealized model; this will translate into a closer approximation to real behavior when scaled up.

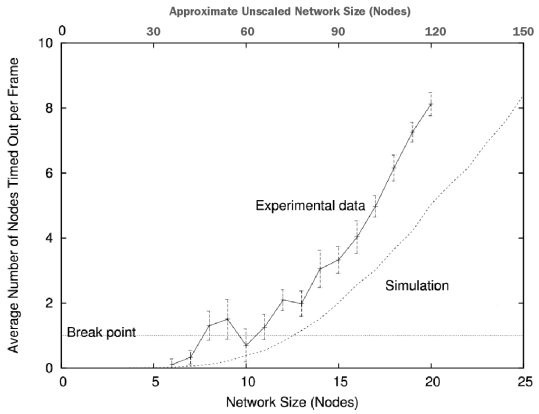

Figure 4: Mean timeouts per frame versus network size for a scaled experiment and one run of a scaled simulation for 0 to 25 nodes.

While our simulation captures the shape of the experiments and the rough magnitude, additional work is required to verify these results with other sets of hardware to control for possible hardware differences. For this reason, the simulation should not be adjusted to fit these specific results more accurately.

7 Using Hardware Effects to Increase Robustness

The fundamental issue discovered in the course of this research is that despite the overall beneficial effect of the hardware imperfections on the behavior of the protocol, the feast and famine dichotomy between particular nodes still limits the supported densities. However, if S-MAC could be modified to expect and work with the hardware effects, even greater network densities could be supported before failure.

We propose a simple modification to S-MAC to recover from hardware clock offset and evaluate it through simulation. An implementation is underway as future work.

7.1 An Offset-Aware MAC

When a node is being deprived because it has a late offset, it will frequently experience offset-related preemptions as described in section 3.4. By comparing the events that occur during an offset-related preemption to the events of a normal preemption, the observed differences between them can be leveraged to allow nodes to determine which kinds of preemptions they are experiencing. This in turn will allow them to infer the direction in which their clock is offset.

The solution to a similar problem with IEEE 802.11 ad-hoc freqency-hopping synchronization described by Huang [3] also recommends that the fastest clock should detect implicitly that it is fast, and correct itself. They discovered the same problem that we encountered, however their focus was on maintaining synchronized clocks, while we focus on improving fairness. Their solution therefore attempts to match the values and rates of all clocks, while we are satisfied with a scheme that converges on long-term fairness while tolerating and even encouraging potentially large short-term discrepencies

A synchronization slot is 2 ms in length. Let deprived node A with late offset choose slot $j > 0$. In a normal preemption, the slot $j - 1$ will be entirely filled with the body of another node's synchronization packet. In an offset-related preemption caused by node A's late offset, only the tail end of what node A considers to be slot $j-1$ will actually be filled. A node B which has an early offset and chooses slot j will begin transmitting at about slot $j - \frac{1}{2}$ by A's reckoning. Figure 5 shows a conceptual diagram of the cases for a node attempting to send in slot $j = 5$. Normal preemption occurs if some other node acquired the channel in an earlier slot ($j = 2$ is shown, labeled "normal preemption"). The case of offset-related preemption occurs when a competitor is slightly early. We can distinguish these cases because with normal preemption the channel is busy at pre-sense time (a half slot early), but not for offset-related preemption.

Therefore, since A carrier senses continuously from the beginning of the frame, A can take a preliminary RSSI reading (*pre-sense*) at slot $j - \frac{1}{2}$. If the channel is busy at this time, A can assume that any preemption is a normal preemption. But if the channel is empty and A is preempted anyway, A can assume that the preemption is offset-related, and that it has a late offset.

A node can then maintain a value called *greediness* which is incremented whenever it successfully transmits a synchronization packet and decremented whenever it experiences an offset-related preemption. If a node's greediness is below a certain threshold, it considers itself to have a late offset and will artificially reduce its lateness by considering each slot to begin 1 ms earlier than it actually does. If a node's greediness is above a certain threshold, it will consider itself to have an early offset and will perform the opposite correction. Using a single counter for both of these mechanisms prevents the feast/famine situation from simply reversing itself, since once a node successfully synchronizes several times consecutively, it will back off.

This algorithm does not attempt to synchronize the nodes' clocks (unlike that of Huang [3]). In fact, the pessimism of the idealized model indicates that synchronized clocks are detrimental to network robustness in this circumstance. Rather, this algorithm attempts to increase fairness amongst nodes so that no node or group of nodes capture the channel at the expense of another group. A similar mechanism for remedying the network capture effect in ethernet is proposed by Ramakrishnan [7] which also seeks to mildly punish nodes which experience repeated success.

In simulation, implementing the mechanism described above with a greediness threshold of 5 to consider oneself to have a late offset and a greediness threshold of -5 to consider oneself to have an early offset increases the number of supported nodes in an unscaled dense S-MAC network by a factor of 2—from 90 to more than 180 nodes. Figure 6 shows a comparison of S-MAC with and without offset correction.

7.2 The Effect of Offset Correction on Density Limitations

According to the idealized model, a full-scale S-MAC network should fail at no fewer than 35 nodes. We

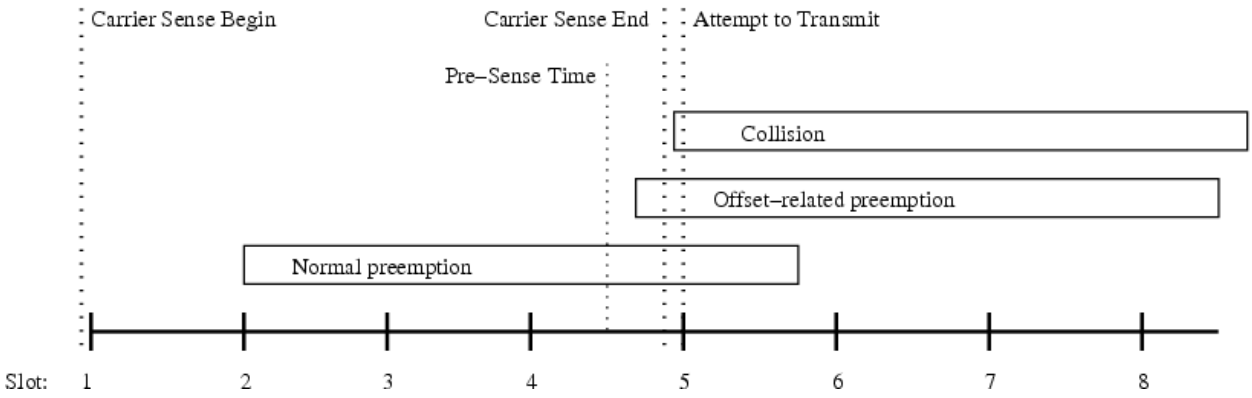

Figure 5: Conceptual diagram of the distinguishing difference between a normal and an offset-related preemption for $j = 5$.

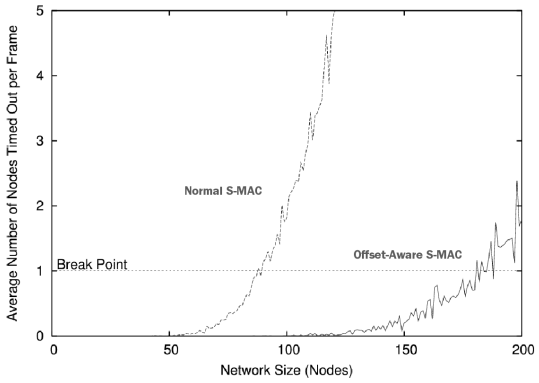

Figure 6: Projected average timeouts per frame versus network size for unscaled simulations with and without offset correction.

consider this a very loose lower bound on reality. According to the simulator (without offset correction), an upper bound, a full-scale S-MAC network should fail by the time it reaches 90 nodes.

While 90 nodes may seem large compared to the number of nodes normally found in a clique of an experimental network today, it is not impressive compared to what S-MAC is theoretically capable of supporting with its default parameters. If it were possible to ensure that every opportunity for synchronization was used while also maintaining a bare minimum of synchronization from all nodes, S-MAC could support many more nodes in a dense clique.

Minimally, each node needs to synchronize only once per timeout interval. And as explained in section 4, each frame presents at most $\lceil \frac{k}{r} \rceil$ opportunities to synchronize. So the theoretical maximum number of nodes that can be supported by S-MAC is given by $\lceil \frac{k}{r} \rceil st$ which, plugging in default parameters, is 324 nodes.

Thus, even at the simulated upper bound, S-MAC in practice supports only about 27% of the nodes that it is theoretically able to support given its current design. The lower limit is due to wasted synchronization slots caused by collisions and deprivation of nodes that are systematically late.

Using offset correction, the failure ceiling should rise considerably. According to the simulated results, an offset-aware S-MAC may survive up to 180 nodes. Offset correction allows S-MAC, in expectation, to double its efficiency and utilize up to 55% of its available capacity.

8 Future Work and Applicability to Other MAC Protocols

Important areas of future work include other approaches to S-MAC stability, general adaptivity in S-MAC, and applicability of these results to other MAC protocols.

We have shown that, with a minor modification to correct for clock-offset and appropriately sized tables, S-MAC can remain synchronized up to networks with more than 100 neighbors. However, the cost of presizing tables for this many neighbors is prohibitive in small-memory motes. Therefore an important step is to avoid multiple schedules, perhaps by adopting the global schedule algorithm [4]. (More recent protocols such as SCP-MAC [13] take this approach.)

Here we have considered only the impact of neighbor table maintenance in S-MAC. The fixed number of slots for RTS/CTS exchanges also poses a problem in active networks. Thus a more general solution might be to adapt network parameters to the current neighborhood size. For synchronization, this would imply adjusting t dynamically. While such dynamic operation is feasible, care must be taken to coordinate any changes to prevent inconsistencies across the network.

In this paper we do not specifically consider other MAC protocols. Most protocols such as 802.11, TDMA protocols (such as Z-MAC [9]), and extensions of S-MAC (such as T-MAC [11]) have similar control algorithms to S-MAC neighborhood maintenance with fixed numbers of slots. We expect that they will encounter similar problems in very dense networks. Other protocols, such as B-MAC [6], operate with much looser synchronization and therefore may encounter fewer direct problems. However, in effect, the long preamble in B-MAC provides synchronization and would become prohibitive in dense networks.

9 Conclusion

This paper provided a preliminary analysis of MAC stability at high network densities. We developed a simple model and a more accurate simulation, validated with experimentation. We have shown that it is possible to accurately predict MAC stability at high densities. In addition, we demonstrated that systematic clock offset caused by hardware variation plays a critical role in MAC stability. If not accounted for, this randomness can result in node deprivation, but when managed with a simple modification to the MAC protocol, we show that hardware randomness can be beneficial to network stability.

We believe the methods described in this paper provide a general guideline for the analysis of MAC protocols at high network densities. Models are an interesting starting point, but useful analysis must take into account hardware effects such as clock offset both for more accurate projections and also for possible improvements to the protocol itself.

Acknowledgments

Thanks to Wei Ye for helpful discussions about the S-MAC protocol and its implementation, and to Affan Sayed for assistance with taking oscilliscope measurements.

References

[1] Harold Abelson, Don Allen, Daniel Coore, Chris Hanson, George Homsy, Thomas F. Knight, Radhika Nagpal, Erik Rauch, Gerald Jay Sussman, and Ron Weiss. Embedding the internet: amorphous computing. *Communications of the ACM*, 43(5):74–82, May 2000.

[2] Javier Gomez and Andrew T. Campbell. A case for variable-range transmission power control in wireless multihop networks. In *Proceedings of the IEEE Infocom*, volume 2, pages 1425–1436, Hong Kong, China, March 2004. IEEE.

[3] Lifei Huang and Ten-Hwang Lai. On the scalability of ieee 802.11 ad hoc networks. In *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 173–182, New York, NY, USA, 2002. ACM Press.

[4] Yuan Li, Wei Ye, and John Heidemann. Energy and latency control in low duty cycle MAC protocols. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, New Orleans, LA, USA, March 2005.

[5] Jeffrey Monks, Vaduvur Bharghavan, and Wen-Mei Hwu. A power controlled multiple access protocol for wireless packet networks. In *Proceedings of the IEEE Infocom*, pages 219–228, Anchorage, Alaska, USA, April 2001. IEEE.

[6] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the Second ACM SenSys Conference*, pages 95–107, Baltimore, MD, USA, November 2004. ACM.

[7] K. K. Ramakrishnan and Henry Yang. The ethernet capture effect: Analysis and solution. In *Proceedings of the 19th IEEE Confrence on Local Computer Networks*, pages 228–240, Minneapolis, Minnesota, USA, October 1994. IEEE.

[8] Ram Ramanathan and Regina Rosales-Hain. Topology control of multihop wireless networks using transmit power adjustment. In *Proceedings of the IEEE Infocom*, volume 2, pages 404–413, Tel Aviv, Israel, March 2000. IEEE.

[9] Injong Rhee, Ajit Warrier, Mahesh Aia, and Jeongki Min. Z-MAC: a hybrid MAC for wirless sensor networks. In *Proceedings of the Third ACM SenSys Conference*, pages 90–101, San Diego, California, USA, November 2005. ACM.

[10] M. Sayrafiezadeh. The birthday problem revisited. *Mathematics Magazine*, 67(3):220–223, June 1994.

[11] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the First ACM SenSys Conference*, pages 171–180, Los Angeles, California, USA, November 2003. ACM.

[12] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *Proceedings of the ACM SenSys Conference*, pages 13–24, Baltimore, Maryland, USA, November 2004. ACM.

[13] Wei Ye and John Heidemann. Ultra-low duty cycle mac with scheduled channel polling. Technical Report ISI-TR-2005-604, USC/Information Sciences Institute, July 2005.

[14] Wei Ye, John Heidemann, and Deborah Estrin. Medium access control with coordinated, adaptive sleeping for wireless sensor networks. *ACM/IEEE Transactions on Networking*, 2003. accepted to appear IEEE/ACM Transactions on Networking; draft available as ISI-TR-567.