Sift: A MAC Protocol for Event-Driven Wireless Sensor Networks*

Kyle Jamieson[1], Hari Balakrishnan[1], and Y.C. Tay[2]

[1] MIT Computer Science and Artificial Intelligence Laboratory,
The Stata Center, 32 Vassar St., Cambridge, MA 02139
jamieson@csail.mit.edu, hari@csail.mit.edu
[2] Department of Computer Science, National University of Singapore,
Kent Ridge 117543, Republic of Singapore
tay@acm.org

Abstract. Nodes in sensor networks often encounter *spatially-correlated contention*, where multiple nodes in the same neighborhood all sense an event they need to transmit information about. Furthermore, in many sensor network applications, it is sufficient if a subset of the nodes that observe the same event report it. We show that traditional carrier-sense multiple access (CSMA) protocols for sensor networks do not handle the first constraint adequately, and do not take advantage of the second property, leading to degraded latency as the network scales in size. We present *Sift*, a medium access control (MAC) protocol for wireless sensor networks designed with the above observations in mind. We show using simulations that as the size of the sensor network scales up to 500 nodes, Sift can offer up to a 7-fold latency reduction compared to other protocols, while maintaining competitive throughput.

1 Introduction

Every shared wireless communication channel needs a medium access control (MAC) protocol to arbitrate access to the channel. Over the past several decades, many MAC protocols have been designed and several are in operation in wireless networks today. While these protocols work well for traditional data workloads, they are inadequate in emerging wireless sensor networks where the nature of data transmissions and application requirements are different. This paper argues that wireless sensor networks require a fresh look at MAC protocol design, and proposes a new protocol that works well in this problem domain by taking advantage of application requirements and data characteristics. We start with an example of a real sensor network.

Machine room monitoring. A fire in a basement machine room of the computer science building triggers a number of redundant temperature and smoke sensors to begin reporting the event. They all simultaneously become backlogged with the sensor reports and

* This material is based upon work supported by the National Science Foundation under Grant Nos. CNS-0205445 and CNS-0520032. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

K. Römer, H. Karl, and F. Mattern (Eds.): EWSN 2006, LNCS 3868, pp. 260–275, 2006.

use a MAC protocol to arbitrate access to the medium. Higher-level applications need some number of event reports that is less than the number of reporting sensors.

From these example, we make the following observations:

1. *Many sensor networks are event-driven and have spatially-correlated contention.* In most sensor networks, multiple sensors are deployed in the same geographic area, usually for fault-tolerance and reliability. In addition to sending periodic observations, when an event of interest happens, the sensing nodes that observe the event send messages reporting the event. The result is *spatially-correlated contention*. Multiple sensors sharing the wireless medium all have messages to send at almost the same time because they all generate messages in response to the same event.
2. *Not all sensing nodes need to report an event.* In sensor network applications such as the machine room example above, not all the nodes that sense an event need to report it. It is enough for a subset of the event reports to reach the data sink.
3. *The density of sensing nodes can quickly change.* In many sensor networks, the size of the set of sensing nodes changes quickly with time, e.g., when a target enters a field of sensors. The potential for sensor nodes to continue decreasing in size [1] leads us to believe that the number of sensing nodes could quickly become very large. As a result, we need a MAC protocol that not only handles spatial correlations, but also adapts well to changes in the number of contending nodes.

These three observations lead to a problem statement for wireless sensor MAC protocol design that is different from classical MAC design. Specifically, in a shared medium where N nodes sense an event and contend to transmit on the channel at the same time, our goal is to design a MAC protocol that minimizes the time taken to send R of these messages without collisions. Notice that when $R = N$, this becomes the throughput-optimization problem that classical MAC protocols are designed for. When $R < N$, what we seek is a protocol that allows the first R winners of the contention protocol to send their messages through as quickly as possible, with the remaining $N - R$ potential transmitters suppressing their messages once R have been sent. In the rest of this paper, we denote the number of nodes that have data to send as N, and the number of reports that the sink needs as R.

At their core, all randomized carrier-sense multiple access (CSMA)-based MAC protocols attempt to adapt to the active population size of contending nodes. Typically, each node maintains a slotted *contention window* with collisions (i.e., unsuccessful transmissions) causing the window to grow in size, and successful transmissions causing it to shrink. Each node transmits data at a slot picked uniformly at random within the current contention window. This approach does not work well when we are interested in the first R of N potential reports, and has problems scaling well when N suddenly grows. The result is degraded response latency.

Our protocol, *Sift*, is based on the intuition that when we are interested in low latency for the first R reports, it is important for the first few successful slots to be contention-free. To tightly bound response latency, we use a *fixed-size* contention window, but a non-uniform, geometrically-increasing probability distribution for picking a transmission slot in the window.

We give theoretical justification for Sift's choice of geometrically-increasing probability distribution and show using simulations that Sift can offer up to a 7-fold latency reduction as the number of sensors in one radio range scales up to 500 nodes. We also show that Sift delivers slightly worse throughput than other CSMA protocols when N is small, and slightly better throughput when N is large. Finally, we describe the theoretically-optimal non-persistent CSMA MAC when one report of each event is enough, and show that Sift's latency approaches optimal.

2 Sift Design

Sift is a non-persistent CSMA wireless MAC protocol. In such protocols, the time immediately after any transmission is divided into CW contention slots, whose duration is usually several orders of magnitude smaller than the time it takes to send a data packet. Immediately after a transmission or collision, each station picks a random contention slot $r \in [1, CW]$. During the contention slots prior to r, each station carrier senses the medium, and aborts or delays its pending transmission if it hears the beginning of another transmission. At contention slot r, the station begins its transmission. If two nodes pick the same slot, they both transmit at the same time, causing a collision. Wireless nodes infer that a collision has occurred by the absence of a link-level acknowledgment. When a collision occurs, most CSMA protocols specify that the colliding nodes double their value of CW. This is known as *binary exponential backoff* (*BEB*). 802.11 [2], B-MAC [3], S-MAC [4], and MACAW [5] are all based on BEB.

By increasing CW, most other CSMA protocols attempt to adapt to the current active population size to make a collision-free transmission more likely. There are two problems with this method. First, it takes time for CW to increase to the right value when the active population (N) becomes large, such as when an event is observed by many sensors after a previously-idle period. Second, if CW is already large (because of traffic congestion that has just subsided) and N is small, then such protocols waste

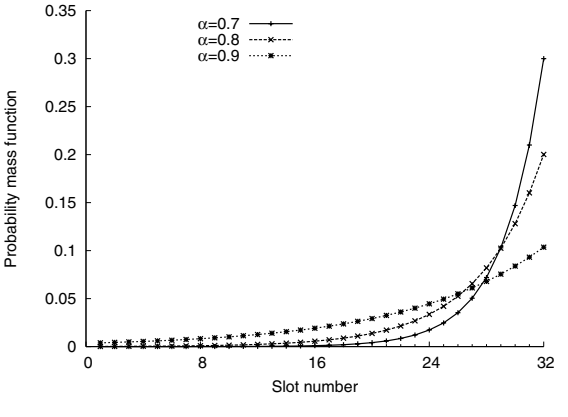

Fig. 1. The probability distribution for the contention slot number that each Sift station chooses. We show various values of α, the parameter of the distribution.

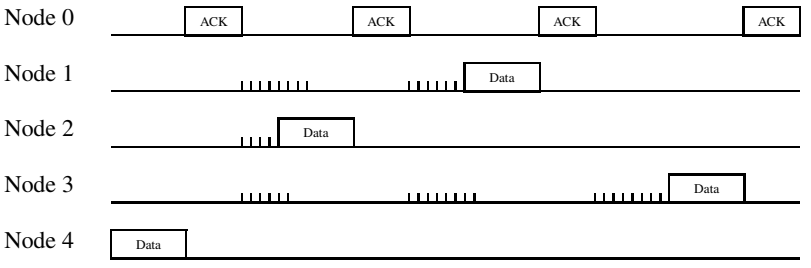

Fig. 2. A timeline of five nodes running the Sift protocol with $N = R = 4$. Nodes 1–4 each send one packet to Node 0. Every time the medium becomes idle, stations re-select a slot at random according to the Sift distribution (Figure 1) before transmitting in that slot. The small bars signify contention, and the number of small bars signifies which slot each station picked.

bandwidth "backing off." Furthermore, CW is usually chosen to ensure that all active nodes get a chance to send their data, whereas we are interested in the collision-free transmission of the first R of N potential reports of some event.

In contrast to previous protocols, Sift uses a small and fixed CW. Of course, nodes can no longer pick contention slots from a uniform distribution, because this would lead to collisions for even moderate values of N. The key difference between Sift and previous CSMA-based wireless MAC protocols is that the probability of picking a slot in this interval is not uniform. Instead, with a carefully-chosen fixed CW and fixed probability distribution, we will show that Sift can perform well in a sensor network.

The following intuition leads us to propose the *geometrically-increasing* probability distribution for picking a contention slot, shown in Figure 1. When N is large, most nodes will choose medium to high slot numbers to transmit (see Figure 1), but a small number will choose low slot numbers, making a collision-free transmission likely in a low slot number. When N is medium, most nodes will choose high-numbered slots, making a collision-free transmission likely in a medium slot number. Finally, when N is small, a collision-free transmission is likely in a high slot number[1]. Thus, for any N, and no matter how fast N changes, a collision-free transmission is likely. We make this intuition precise in Section 2.1.

Figure 2 shows an example run of the Sift MAC protocol. Note that when the transmission or collision finishes, all competing Sift nodes select *new* random contention slots, and repeat the process of contending over the fixed contention window.

In the rest of this section we describe Sift's probability distribution and compare it to an optimal (for $R = 1$) non-persistent CSMA. We then give a formal protocol specification, and qualitatively compare Sift to other contention window-based CSMA protocols.

2.1 The Sift Probability Distribution

Suppose each sensor picks a slot $r \in [1, CW]$ with probability p_r. We say that slot r is *silent* if no sensor chooses that slot, and there is a *collision* if more than one sensor

[1] This is the motivation behind the name *Sift*: the non-uniform probability distribution "sifts" the (collision-free) winners from the entire contending set of nodes.

chooses that slot. Also, a sensor *wins in slot* r if it is the only one to choose slot r, and all others choose later slots. Finally, there is *success* if some sensor wins some slot in $[1, CW]$.

Sift uses the truncated, increasing geometric distribution

$$p_r = \frac{(1-\alpha)\alpha^{CW}}{1-\alpha^{CW}} \cdot \alpha^{-r} \quad \text{for } r = 1, \ldots, CW, \tag{1}$$

where $0 < \alpha < 1$ is a parameter. For these values of α, p_r increases exponentially with r, so the later slots have higher probability.

To motivate this choice, view each sensor's choice of which slot to pick as a decision procedure with CW stages. Each node starts in stage 1 with some overestimate N_1 of N and chooses slot 1 with a small probability.[2] If no sensor chooses slot 1, that is an indication that N_1 is an overestimate of N, so each node updates its guess of the population size by decreasing N_1 to N_2, and proceeds to choose slot 2 with a different probability in stage 2. If slot 2 is also silent, this guess is reduced to N_3 in stage 3, and so on; in general, N_r is the updated guess after there is silence in slots $1, \ldots, r-1$. In previous work [6], we have shown that a near-optimal choice of α for a wide range of population sizes is $\alpha = N_1^{-\frac{1}{CW-1}}$.

The points in Figure 3 plot the result of an experiment in which N sensors choose slots using the distribution in Equation 1 with $\alpha = 512^{-\frac{1}{31}} \approx 0.818$. Each point in the graph represents one run with N sensors. Note that although we engineered the Sift probability distribution for a maximum number of sensors $N_1 = 512$, performance degrades gracefully when the true number of contending stations exceeds 512. This degradation happens because the first slot starts to get picked by more than one sensor, resulting in a collision. We ran the same simulation with α set to various values in the range $[0.7, 0.9]$. Our results verified that we had chosen the correct α, and that over this range, the success rate is not sensitive to the exact choice of α.

Figure 3 also shows that although the sensors do not know N and use a fixed distribution p_r, the probability of a successful transmission is constantly high for a large range of N. In the next section, we will see that this probability of success is in fact close to the maximum that is achievable even if the sensors knew N and used a distribution tuned for N. We emphasize that we introduced N_r and p'_r here for explanatory purposes only, as a way of understanding our choice of p_r. In particular, nodes running Sift do not maintain an explicit estimate of N_r.

2.2 Comparison with an Optimal Protocol

Suppose each contending station had perfect knowledge of the true number of contending stations at the instant it started contending for the shared medium, and picked a contention slot in which to transmit at the beginning of the contention period, with no

[2] N_1 is a fixed parameter that defines the maximum population size Sift is designed for. All practical MACs have such a parameter; for example 802.11 limited the maximum contention window size to 1024 for commodity hardware at the time this paper was written. In Section 2.2, we show that above this population size, Sift's performance degrades gracefully.

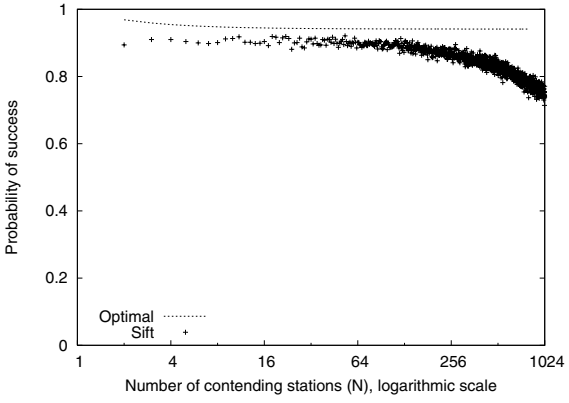

Fig. 3. A comparison between Sift with $\alpha = 0.818$ and $CW = 32$, and an optimal protocol, with $CW = 32$. The optimal protocol has full knowledge of N; Sift has no knowledge of N. The Sift distribution shown above was engineered for a maximum value of $N = 512$ nodes, but its performance degrades gracefully when N exceeds that figure.

other information provided to it during the contention period.[3] In related work [6], we derive the distribution p^* that optimizes the probability of a successful transmission.

Figure 3 shows the success probability of the Sift distribution as well as the theoretical success probability of the optimal distribution. When $R = 1$, the Sift distribution (which does not know N) performs almost as well as the optimal distribution (which needs to know N). As we argued in Section 1, it is most often the case that N is unknown and hard to predict.

The RTS/CTS Exchange. For large packet sizes (those above a tunable threshold), Sift uses the RTS/CTS exchange in the same way as IEEE 802.11 [2]. Instead of using the Sift distribution to compete on data packets, we use it to compete on sending the RTS packet. Since sensor network workloads mostly contain short packets, we evaluate the Sift's performance in Section 3 and 802.11 with RTS/CTS disabled, sending short data packets. In Section 3.4 we run some experiments with RTS/CTS enabled, sending large data packets.

Hidden Terminals. Modern spread-spectrum radios have a carrier-sensing range approximately twice that of their transmission range [2, 7], making it more likely that a node will carrier-sense a transmission that can interfere at the receiver of its transmission. This lessens the frequency of hidden terminals. For large packets, Sift uses the RTS/CTS exchange to avoid collisions between hidden terminals. In the case of collisions between small data packets among hidden terminals, senders can arbitrate as CODA [8] proposes, or can vary their transmit phases with respect to one other to avoid collisions [9]. We evaluate Sift under hidden terminal conditions in Section 3.5.

[3] These conditions exclude non-contention-window-based protocols like tree-splitting contention resolution. We address such protocols in related work [6].

Implementing Suppression. In the introduction, we described a workload in which sensors suppress their event reports after some number of reports R have been sent. In some scenarios, such as the last hop to a base station, this suppression is not hard to implement: sensors listen to the base station for R acknowledgment packets (ACKs) from data packets delivered to the base station. In general, when not at the last hop to a base station, sensors listen for R events timestamped within some time interval away from the event of interest before suppressing their event report.

2.3 Exploring the CSMA Design Space

Current sensor network designs (such as B-MAC [3], the MAC layer of TinyOS[4]) use a fixed-window CSMA protocol, choosing contention slots uniformly at random. The advantage of this design choice is simplicity, and good performance under most practical sensor network deployment scenarios. The disadvantage of this design choice is a lack of scalability under highly-correlated traffic or large numbers of sensor nodes.

Bharghavan *et al.* proposed MACAW [5], a MAC protocol for wireless local-area networks. MACAW uses BEB (described at the beginning of Section 2), and so without some way to share information about the state of the wireless medium, MACAW would suffer from the well-known Ethernet capture problem: a station that just transmitted resets its contention window to the minimum value, and is thus more likely to transmit again in subsequent competitions. MACAW's solution to this belongs to a class of techniques that we term *shared learning*. Stations *copy* the CW value of a station that transmits to their own CW value, and modify BEB so that instead of resetting CW after a successful transmission, decreases it linearly (a multiplicative increase, linear decrease policy).

Instead of shared learning, 802.11 [2] uses *memory* to solve the fairness problem. When stations begin to compete, they set a countdown timer to a random value picked uniformly from the contention window CW. When the medium becomes busy, the station pauses the countdown timer. When the medium becomes idle and the station wants to compete again, 802.11 resumes its countdown timer. When the countdown timer expires, the station begins its transmission.

In 802.11, a station that successfully transmits resets its CW to a small, fixed minimum value of CW. Consequently, the station has to rediscover the correct CW, wasting some bandwidth.

Table 1. Some design parameters in the contention window-based CSMA space. Sift requires neither shared learning, a variable-sized contention window, nor memory to perform well.

Protocol	Contention window	Shared learning?	Memory?	Distribution
BEB	Variable	No	No	Uniform
802.11	Variable	No	Yes	Uniform
MACAW	Variable	Yes	No	Uniform
802.11/copy	Variable	Yes	Yes	Uniform
B-MAC, S-MAC	Fixed	No	No	Uniform
Sift	**Fixed**	**No**	**No**	**Reverse-exponential**

[4] See `http://tinyos.net`.

One might think that shared learning could help the problem of high rate of change of N with respect to time. The spurious intuition behind this is that when a node is successful in its transmission, it might have found the correct value of CW for all nodes to use. 802.11 with shared learning, which we term *802.11/copy*, still suffers when N increases quickly. We substantiate this claim in Section 3.

Table 1 summarizes the design parameters we have reviewed. From the table, it is clear that Sift explores a novel region of the contention window-based MAC design space. We now show that this particular point in the design space results in good performance in sensor networks with respect to throughput, latency, and fairness.

3 Performance Evaluation

In our experiments, we compare Sift configured with $CW = 32$ and $\alpha = 0.818$ to 802.11 and 802.11/copy (defined in Section 2.3). We choose the 802.11 family because it is a practical CSMA protocol whose mechanism for adapting to the number of transmitting stations (BEB) has been included, unmodified, in several proposals for the MAC layer of a sensor network [4, 9].

We run experiments using version 2.1b9 of the *ns-2* [7] network simulator, with all nodes within range of a common base station. We modify all the MACs in our experiments to perform suppression: if a sensor hears R acknowledgments for motion event $\mathcal{E}$ from the base station, it suppresses its report of $\mathcal{E}$ and removes $\mathcal{E}$'s packet from its transmit queue. For experiments with small data packets (40 bytes), we compare Sift, 802.11 *without* RTS/CTS, and 802.11/copy *without* RTS/CTS. For the fairness experiments in Section 3.4, where data packets are 1500 bytes long, we enable RTS/CTS for both Sift and the 802.11 protocols. All experimental results average 20 runs using different random seeds for each run, except the fairness experiments in Section 3.4 which average 5 runs.

3.1 Event-Based Workloads

Constant-bit-rate (CBR) or TCP flows do not suffice to evaluate protocols for sensor networks, because they capture neither the burstiness inherent in the network, nor some underlying physical process that the network should be sensing. We therefore propose two event-based workloads to evaluate our design.

Trace-Driven Event Workload. We model a sensor network that detects the presence of people or cars in a region of a busy street. Rather than deploying this sensor network, we acquire video from a camera pointed at a busy street and log motion events to a database. This data captures the physical process of person and car inter-arrival times on the street. We call this the *trace-driven event* workload.

To run an experiment with this trace-driven workload, we create an *ns-2* scenario where sensors are placed uniformly at random on a two-dimensional plane. At the time given by each motion event in the database, all sensors within d_{report} meters of the location of that event send a 40 byte report packet, suppressing their reports after R have been sent.

Constant-Rate Event Workload. In some experiments, we measure the network not in the steady-state, but in a dynamic situation where the network has quiesced, then N nodes sense an event and report that event, suppressing their reports after R have been sent. Event reports are 40 bytes in size. We call this the *constant-rate event* workload.

3.2 Latency Experiments

We begin by evaluating latency under the constant-rate event workload. To capture varying propagation delays in the environment, variations between sensor electronics, and uncertainty in software system delays on the sensor nodes themselves, we add a random delay in $[0, 1]$ ms to the time that each sensor sends its event report. We measure the time for the base station to receive the first, median, and 90th percentile event report. We plot these times as a function of N.

Figure 4 shows the results of this experiment. When N is small, the minimum 802.11 contention window size is large enough to quickly resolve contention between the nodes. As N grows, however, the 802.11 and 802.11/copy contention window needs to grow before even one event report can be successfully sent (see the bottom of the error bars in Figure 4), while Sift's fixed contention window resolves contention in constant time with respect to N. Turning to the median and 90th percentile event reporting times, we see that Sift also improves latency for these measures as well, up to $N = 256$. This is primarily due to Sift's improvement in the first event reporting time, but it also shows that Sift can deliver enough throughput to keep up with 802.11 and 802.11/copy as it sends subsequent event reports.

802.11/copy does not improve performance much because some stations transmit before they have estimated the optimal CW value, broadcasting values of CW that are too low. As a result, CW cannot increase quickly enough when N is large. Sift does not need any time to adapt to large N, and so performs well over a large range of N. Figure 4 shows that as N increases, Sift achieves a seven-fold latency reduction over 802.11.

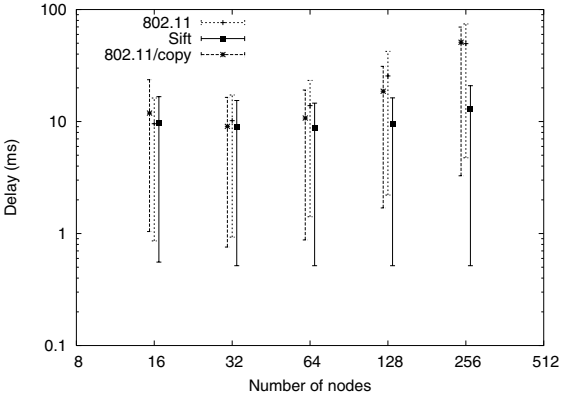

Fig. 4. Latency as a function of N (number of sensors reporting an event). 16 reports are required ($R = 16$). We show the time that the first (bottom of error bar), median (point in error bar), and 90th percentile (top of error bar) event report was received at the base station. This experiment uses the constant-rate event workload, and all three protocols use suppression.

3.3 Throughput Experiments

We now compare the throughput of Sift, 802.11, and 802.11/copy under a variety of workloads that saturate the capacity of the wireless medium.

Trace-Driven Events. Using our trace-driven workload, we measure the time each protocol takes to deliver R reports for each motion event, varying R. N also varies, depending on how many of the 128 total nodes are within range of each motion event, but averages 100. Since the traffic pattern is bursty, when R grows (and the number of reports suppressed shrinks), we quickly reach the capacity of the medium. When this happens, interface queues at the senders start to build up, and latency sharply increases. To examine the capacity of the medium using Sift versus using 802.11, we increased the upper bound on the interface queue length by an order of magnitude, from 50 packets to 500 packets. We then measured the latency to receive R events for Sift, 802.11, and 802.11/copy. Figure 5 shows the results of this experiment. As expected, when R is small, Sift has lower latency than either 802.11 or 802.11/copy, because it can resolve contention faster than a BEB protocol. Furthermore, noting the position of the knee of each curve, Sift can continue delivering low-latency events for higher values of R because it can deliver higher throughput under the varying values of N that this workload generates.

Constant-Rate Events. Now we measure the time it takes to receive R events when N is fixed at 128. Figure 6 shows that Sift achieves better throughput than 802.11 under this workload. The reason for this is again that Sift does not have to track the sudden change in N like BEB does.

The Sift Performance Space. In Figure 7, we explore the Sift performance space when we vary both N and R. Consider first the five bottom-most curves ($R = 1, 2, 4, 8, 16$)

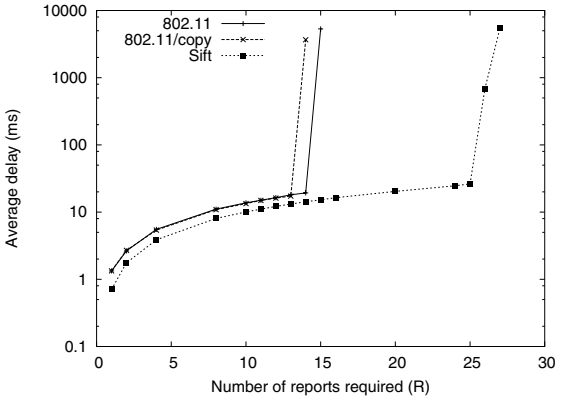

Fig. 5. Average delay as a function of R (number of reports required) for each camera motion event. The sensor range in this experiment is fixed at 20 meters ($d_{\text{report}} = 20$), and there are 128 nodes in this experiment. N is a function of event location, as explained in the text. This experiment uses the trace-driven event workload, and all three protocols use suppression.

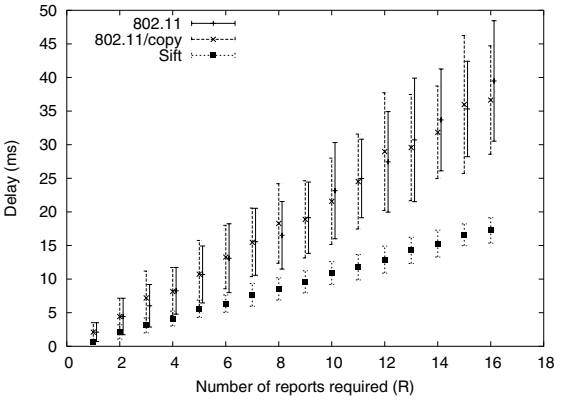

Fig. 6. Average and standard deviation latency to receive R event reports when 128 sensors report an event ($N = 128$). All sensors detect the event at the same time. This experiment uses the constant-rate event workload, and all three protocols use suppression.

with zero slope. They show that no matter how many stations report an event (N), Sift can deliver $R = 1, 2, 4, 8$, or 16 messages with a small, constant latency. Now consider the remaining curves ($R = 24, 32, 64$) in Figure 7, which have a non-zero slope. They show that once R becomes greater than or equal to 24 messages, Sift requires more time to deliver R messages as N grows. Thus to a point, Sift scales well *simultaneously* with respect to R and N.

3.4 Fairness Experiments

We now examine whether Sift fairly allocates bandwidth between stations. It has been shown that 802.11 does not, but that minor changes to 802.11 can yield a fair

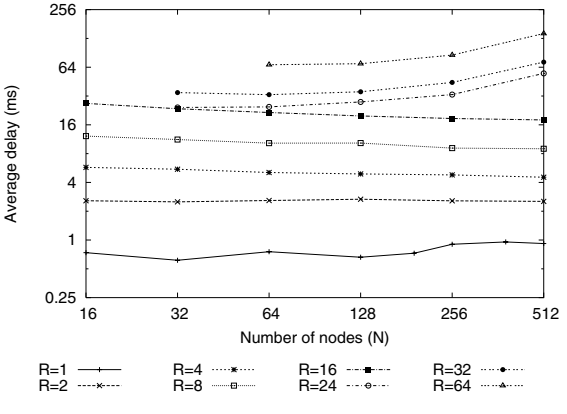

Fig. 7. Average delay of Sift event reports as a function of N (number of sensors reporting an event). We show curves for various values of R (number of reports required). Note that $R \leq N$. This experiment uses the constant-rate event workload with suppression.

protocol [10]. We duplicate the experimental setup given by the authors of the distributed fair scheduling (DFS) protocol [10]. We place some even number of nodes in the same radio range, and set up a traffic pattern where each node is either a traffic source or a traffic sink. The packet size is 1500 bytes, and the RTS/CTS exchange is enabled for both 802.11 and Sift. We ensure that each node is backlogged so that the offered load exceeds the available wireless capacity.

Figure 8 shows the throughput achieved by each node in six seconds as a function of the node number. Note that as expected, 802.11/copy outperforms 802.11 in terms of fairness. Also notice that Sift outperforms 802.11 in terms of fairness. Sift does not in fact achieve a perfectly-fair bandwidth allocation. We expect that this is not a major issue, since sensor networks will contain many redundant nodes reporting similar observations about the environment. However, due to the simplicity of Sift, we expect that a similar approach to DFS could be applied to Sift should fairness become an issue.

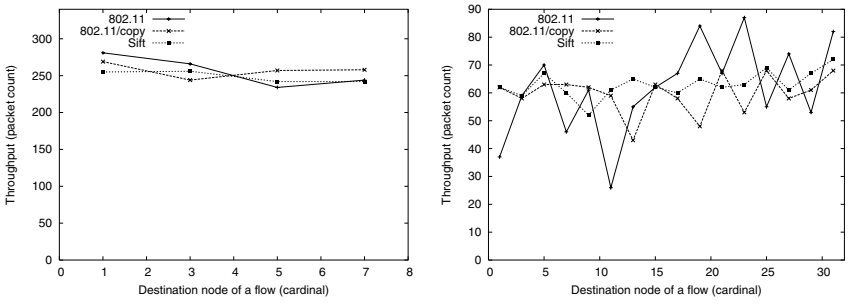

Fig. 8. Fairness comparison of 802.11 and Sift. Left: eight nodes; right: 32 nodes. In each experiment there are half as many flows as there are nodes. This experiment uses a CBR workload, without suppression. RTS/CTS is enabled for all three protocols.

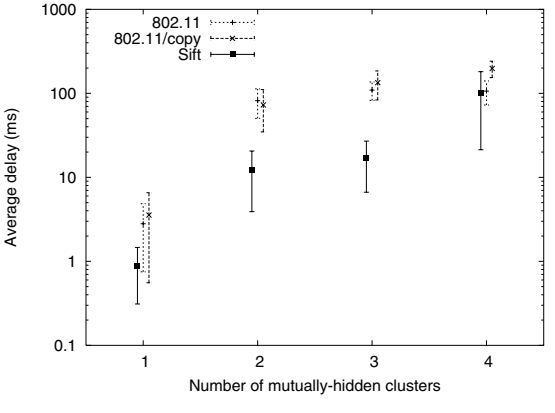

Fig. 9. Latency comparison between Sift and 802.11 in the presence of varying numbers of hidden terminals. This experiment uses an event-based workload with suppression. RTS/CTS is disabled, and each packet is 40 bytes in length.

3.5 Hidden Terminal Experiments

We now compare Sift with 802.11 and 802.11/copy in the presence of hidden terminals. In this experiment, we arrange $N = 128$ nodes in closely-spaced clusters around the base station to which they send event reports. Nodes in each cluster can carrier-sense each others' transmissions, and defer and suppress accordingly. Nodes in separate clusters cannot carrier-sense each others' transmissions: they are hidden terminals with respect to each other. We vary the number of clusters around the base station, and measure the time to receive $R = 1$ report. Figure 9 shows that as the number of hidden terminals increases, latency increases due to a significantly-increased number of collisions. Sift performs better than 802.11 in hidden terminal situations because it does not incur the penalty of contention-window doubling when a collision occurs.

4 Related Work

We compared Sift to 802.11, B-MAC, and MACAW in Section 2.3. We now review more related work.

There have been a number of proposals [11–14] for controlling the flow of information in a sensor network at the application layer. While these proposals are essential, they are orthogonal to the choice of MAC layer, and that choosing an appropriate MAC is important for the performance of a sensor network.

Cai *et al.* [15] propose CSMA with a truncated polynomial distribution over a fixed contention window. There are several significant differences between their proposal and Sift. First, Sift uses the exponential distribution, which is close to optimal over all possible distributions, as described in Section 2.2. Furthermore, Cai *et al.* optimize only over the polynomial distributions, not over all possible distributions. Finally, Sift was designed and evaluated in an event-based workload (see Section 3), while Cai *et al.* evaluate their proposal using a Poissonian workload.

Like Sift, the HIPERLAN standard [16] for wireless LANs uses a truncated geometric probability distribution in the "elimination" phase of its contention protocol. Cho *et al.* [17] describe and analyze HIPERLAN's MAC protocol in detail. Sift uses traditional CSMA, where immediately following a busy channel, the first station to break the silence wins access to the medium. In contrast, HIPERLAN stations transmit noise bursts of varying length after the medium becomes idle, and the station that ceases its noise burst *last* wins access to the medium. Sift compares favorably with HIPERLAN for two reasons. HIPERLAN's noise bursts raise the overall noise floor of the network when there are many stations, and consume more power than listening for the same amount of time on most radio hardware.

Mowafi and Ephremides [18] propose Probabilistic Time Division (PTD), a TDMA-like scheme in which stations transmit in each TDMA slot with a given probability. Each station chooses one TDMA slot in each round with a fixed probability a. By tuning a, PTD achieves a compromise between TDMA and pure random access. Our proposal differs from PTD because we compute an optimal probability distribution on *contention slots*, which in practice are several orders of magnitude smaller than TDMA data slots.

Rhee *et al.* propose Z-MAC [19], a MAC for sensor networks that combines the strengths of TDMA and CSMA. Sift stands out from Z-MAC because Sift's probability distribution reduces the likelihood of collisions compared to CSMA's uniform distribution. Z-MAC is one of many examples of a MAC protocol that could incorporate Sift's probability distribution to improve performance.

Tan and Guttag [20] demonstrate that 802.11 nodes use globally-inefficient transmission strategies that lead to degraded aggregate throughput. They propose changes to 802.11's backoff window that increase network capacity. Sift has the orthogonal goal of minimizing response latency of a wireless network.

Woo and Culler [9] compare the performance of various contention window-based MAC schemes, varying carrier sense time, contention window size increase/decrease policies, and transmission deferral policies. All of their protocols use contention windows with the uniform distribution. They find that CSMA schemes with a fixed-size window are the most energy-efficient, since nodes spend the least time listening. This further motivates the case for Sift, because Sift uses a fixed-size contention window. Woo and Culler also find that fixed-size contention window protocols perform well in terms of throughput.

S-MAC [4] is a MAC protocol designed for saving energy in sensor networks. It uses periodic listen and sleep, the collision avoidance facilities of 802.11, and overhearing avoidance to reduce energy consumption. LEACH [21] is designed for sensor networks where an end-user wants to remotely monitor the environment. It includes distributed cluster formation, local processing to reduce global communication, and randomized rotation of the cluster-heads to extend system lifetime. PAMAS [22] reduces energy consumption by powering off nodes when they are about to overhear a transmission from one of their neighbors. While S-MAC, LEACH, and PAMAS govern medium-access to some degree they do not address the contention portion of a medium-access protocol. Since Sift is a CSMA protocol, it can be implemented concurrently with these protocols.

There have also been a number of proposals [23–25] for topology-control in wireless networks. Although their goal is energy savings, if topology formation protocols could be adapted to take into account the number of sensor reports required, it might be possible to provide an alternate solution to our problem; we leave this idea as future work. We note that Sift can be used as a building block in the underlying MAC to arbitrate access between the large numbers of nodes that need to rendezvous at the same time and elect coordinators.

5 Conclusion

We have presented Sift, a MAC protocol for wireless sensor networks that performs well when spatially-correlated contention occurs and adapts well to sudden changes in the number of sensors that are trying to send data. Sift is ideal for sensor networks, where it is often sufficient that any R of N sensors that observe an event report it, with the remaining nodes suppressing their transmissions. The key idea in Sift is to use a geometrically-increasing probability distribution within a fixed-size contention window, rather than varying the window size as in many traditional MAC protocols.

Using trace-driven experiments, we have shown that Sift outperforms 802.11 and other BEB-based protocols both when the ratio R/N is low, and when both N and R are large. For $R = 1$, we have identified the optimal non-persistent CSMA protocol, and shown that Sift's performance is close to optimal.

References

1. Kahn, J., Katz, R., Pister, K.: Mobile Networking for Smart Dust. In: Proc. of the ACM MOBICOM Conf., Seattle, WA (1999) 271–278
2. IEEE 802.11 Standard: Wireless LAN Medium Access Control and Physical Layer Specifications (1999)
3. Polastre, J., Hill, J., Culler, D.: Versatile Low Power Media Access for Wireless Sensor Networks. In: Proc. of the ACM SenSys Conf., Baltimore, MD (2004) 95–107
4. Ye, W., Heidemann, J., Estrin, D.: An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In: Proc. of the IEEE INFOCOM Conf., New York, NY (2002) 1567–1576
5. Bharghavan, V.: MACAW: A Media Access Protocol for Wireless LANs. In: Proc. of the ACM SIGCOMM Conf., London, UK (1994) 212–225
6. Jamieson, K., Tay, Y.C., Balakrishnan, H.: Sift: a MAC Protocol for Event-Driven Wireless Sensor Networks. Technical Report MIT-LCS-TR-894, Massachusetts Institute of Technology (2003)
7. USC ISI: ns-2 Notes and Documentation (2002) http://www.isi.edu/nsnam/ns.
8. Wan, C.Y., Eisenmen, S., Campbell, A.: CODA: Congestion Control in Sensor Networks. In: Proc. of the ACM Sensys Conf., Los Angeles, CA (2003) 266–279
9. Woo, A., Culler, D.: A Transmission Control Scheme for Media Access in Sensor Networks. In: Proc. of the ACM MOBICOM Conf., Rome, Italy (2001) 221–235
10. Vaidya, N., Bahl, V., Gupta, S.: Distributed Fair Scheduling in a Wireless LAN. In: Proc. of the ACM MOBICOM Conf., Boston, MA (2000) 167–178
11. Intanagonwiwat, C., Govindan, R., Estrin, D.: Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In: Proc. of the ACM MOBICOM Conf., Boston, MA (2000) 56–67
12. Madden, S., Franklin, M., Hellerstein, J., Hong, W.: TAG: a Tiny AGregation Service for Ad-Hoc Sensor Networks. In: Proc. of the USENIX OSDI Symp., Boston, MA (2002) 131–146
13. Ye, F., Luo, H., Cheng, J., Lu, S., Zhang, L.: A Two-Tier Data Dissemination Model for Large-Scale Wireless Sensor Networks. In: Proc. of the ACM MOBICOM Conf., Atlanta, GA (2002) 148–159
14. Ratnasamy, S., Karp, B., Yin, L., Yu, F., Estrin, D., Govindan, R., Shenker, S.: GHT: A Geographic Hash Table for Data-Centric Storage. In: Proc. of the ACM WSNA Workshop, Atlanta, GA (2002) 78–87
15. Cai, Z., Lu, M., Wang, X.: Randomized Broadcast Channel Access Algorithms for Ad Hoc Networks. In: Proc. of the IEEE Intl. Conf. on Parallel Processing. (2002) 151–158
16. European Telecommunication Standard: HIgh PErformance Radio Local Area Network (HIPERLAN) Type 1; Functional Specification (1996)
17. Cho, K.O., Shin, H.C., Lee, J.K.: Performance Analysis of HIPERLAN Channel Access Control Protocol. Proc. of the ICICE Trans. on Comm. **E85-B** (2002) 2044–2052
18. Mowafi, O., Ephremides, A.: Analysis of a Hybrid Access Scheme for Buffered Users—Probabilistic Time Division. IEEE Trans. on Software Engineering **8** (1982) 52–60
19. Rhee, I., Warrier, A., Aia, M., Min, J.: Z-MAC: A Hybrid MAC for Wireless Sensor Networks. In: Proc. of the ACM SenSys Conf., San Diego, CA (2005) 90–101

20. Tan, G., Guttag, J.: The 802.11 MAC Protocol Leads to Inefficient Equilibria. In: Proc. of the IEEE INFOCOM Conf., Miami, FL (2005) 1–11
21. Heinzelman, W., Chandrakasan, A., Balakrishnan, H.: Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In: Proc. of the 33rd Hawaii International Conf. on System Sciences (HICSS). (2000)
22. Singh, S., Woo, M., Raghavendra, C.: Power-Aware Routing in Mobile Ad Hoc Networks. In: Proc. of the ACM MOBICOM Conf., Dallas, TX (1998) 181–190
23. Chen, B., Jamieson, K., Balakrishnan, H., Morris, R.: Span: An Energy-Efficient Coordination Algorithm for Topology Maintainence in Ad Hoc Wireless Networks. In: Proc. of the ACM MOBICOM Conf., Rome, Italy (2001) 85–96
24. Wattenhofer, R., Li, L., Bahl, V., Wang, Y.M.: Distributed Topology Control for Wireless Multihop Ad-hoc Networks. In: Proc. of the IEEE INFOCOM Conf., Anchorage, AK (2001) 1370–1379
25. Xu, Y., Heidemann, J., Estrin, D.: Geography-Informed Energy Conservation for Ad Hoc Routing. In: Proc. of the ACM MOBICOM Conf., Rome, Italy (2001) 70–84