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

Kyle Jamieson, Hari Balakrishnan
MIT Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
{jamieson, hari}@lcs.mit.edu

Y. C. Tay
Department of Mathematics
National University of Singapore
tay@acm.org

May 1, 2003

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 like 802.11 do not handle the first constraint adequately, and do not take advantage of the second property, leading to degraded latency and throughput as the network scales in size.

We present *Sift*, a medium access protocol for wireless sensor networks designed with the above observations in mind. Sift is a randomized CSMA protocol, but unlike previous protocols, does not use a time-varying contention window from which a node randomly picks a transmission slot. Rather, to reduce the latency for the delivery of event reports, Sift uses a fixed-size contention window and a carefully-chosen, non-uniform probability distribution of transmitting in each slot within the window. We show using simulations that Sift can offer up to a 7-fold latency reduction compared to 802.11 as the size of the sensor network scales up to 500 nodes. We then analytically prove bounds on the best latency achievable by a decentralized CSMA-based MAC protocol for sensor networks where one report of each event is enough, and show that Sift comes close to meeting this bound.

1 Introduction

Every shared wireless communication channel needs a medium-access control (MAC) protocol by which nodes contend for the channel and eventually transmit without collisions. 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 are not the first researchers to argue for new MAC protocols in the sensor network domain, where several researchers have argued for protocols that conserve energy better than traditional protocols [15, 32]. In addition to energy there are several reasons for rethinking MAC design for sensor networks. Our work is motivated by the following observations:

1. *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 external events.

2. *Not all sensing nodes need to report an event.* In many sensor applications, not all the nodes that sense an event need to report it. Specifically, we find that many applications are designed to have every sensing node send a message, but it is enough for a subset of these messages to reach the data sink.

3. *Time-varying density of sensing nodes.* In many sensor networks, the size of the set of sensing nodes changes with time, e.g., when a target enters a field of sensors. Future sensor networks like Smart Dust [17] lead us to believe that the number of sensing nodes could very quickly become very large. As a result, we need a MAC protocol that not only handles spatial correlations (Observation 1), 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 $R \leq N$ 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. We discuss possible ways of implementing this suppression in Section 2.3.2.

There have been a number of proposals [14, 16, 19, 20, 30, 31] for controlling the flow of information in a sensor network at the application layer. We believe that these types of protocols are extremely useful for sensor networks. At the heart of these protocols, however, the foregoing observations still hold. Furthermore, solving our problem at the application layer with a less-intelligent MAC has serious performance issues, as we will show in our experimental evaluation. We believe that the Sift MAC will be a useful lower-level building block for improving the performance of existing higher-level sensor network protocols, and building new ones.

At their core, all randomized CSMA-based MAC protocols attempt to adapt to the active population size of contending nodes. Typically, this is done using a time-varying “contention window” of a number of slots maintained by each node, with collisions (i.e., unsuccessful transmissions) causing the window to increase and successful transmissions causing it to decrease. Each node transmits data in 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 probability distribution for picking a transmission slot in the window. We find that a distribution that multiplicatively increases the probability of picking a later slot in the fixed window relative to an earlier slot works well.

We give theoretical justification for Sift’s choice of non-uniform 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 analytically prove bounds on the best latency achievable by a decentralized CSMA-based MAC protocol for sensor networks where one report of each event is enough, and show that Sift comes close to meeting this bound. Interestingly, we find that Sift is more scalable than 802.11 and beats it in terms of throughput even when $R = N$ for large N.

The rest of this paper is organized as follows. Section 2 presents the details of Sift and explains why it works well.

Section 3 presents the results of several simulation experiments, including trace-driven experiments. Section 4 discusses related work on MAC protocols in wireless networks. Finally, we conclude with a discussion of the advantages and limitations of Sift in Section 5.

2 Sift Design

Sift is a contention window-based MAC protocol. In all contention window-based protocols, each node picks a random contention slot in $[1, CW]$ to transmit. If two nodes pick the same slot, they will both transmit and cause a collision. When this happens, most protocols specify that the colliding nodes multiplicatively increase their value of CW.

By varying CW, most contention window-based protocols attempt to scale with the current active population size in an attempt to maximize the number of collision-free transmissions without incurring long latencies. There are two problems with this method. First, it takes time for CW to correctly adapt to the right value when the active population is large, as might happen when an event is observed by many sensors after a previous idle period. Second, if CW is already large and an event is observed by only a small number of sensors, then the latency to report the event is high. Furthermore, CW is usually chosen to ensure that all active nodes get a chance to send if they pick values uniformly at random, whereas we are more interested in the collision-free transmission of the first R of N potential reports.

In contrast to previous protocols, Sift uses a small and fixed CW of 32 slots, where each slot lasts on the order of tens of microseconds. 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 protocols like 802.11 is that the probability of picking a slot in this interval is not uniform.

The following intuition leads us to propose a geometrically-increasing probability distribution for picking a contention slot. This distribution is shown in Figure 1. In the next section we formally define the distribution and discuss how to choose the parameter of the distribution α.

Nodes running Sift compete for any slot $r \in [1, CW]$ based on a shared *belief* of the current population size N, which changes after every slot in which no node transmits. This belief starts off at some large value, with a correspondingly-small per-node transmission probability. If no node starts to transmit in the first slot, then each node reduces its belief of the number of competing nodes by multiplicatively increasing its transmission probability for the second slot. This process is repeated in every slot, allowing for the competition to happen at geometrically-decreasing possible values of N *all* in the same small total number of contention slots[1] CW. The result is that Sift enables the winner to be chosen rapidly across a wide range of potential population sizes without incurring long latency due to collisions.

If exactly one node happens to pick some contention slot, it will start to transmit in that slot. When its transmission finishes, all other competing Sift nodes select *new* random contention slots, and repeat the process of backing off over the fixed contention window. The same process happens if two or more stations happen to pick the same contention slot, resulting in a collision. This is shown graphically in Figure 2.

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

2.1 Sift backoff probability distribution

We now formalize our intuition about the shape of the Sift backoff probability distribution. Suppose each sensor picks a slot $r \in [1, CW]$ using a non-uniform probability p_r. We say slot r is **silent** if no sensor chooses that slot,

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

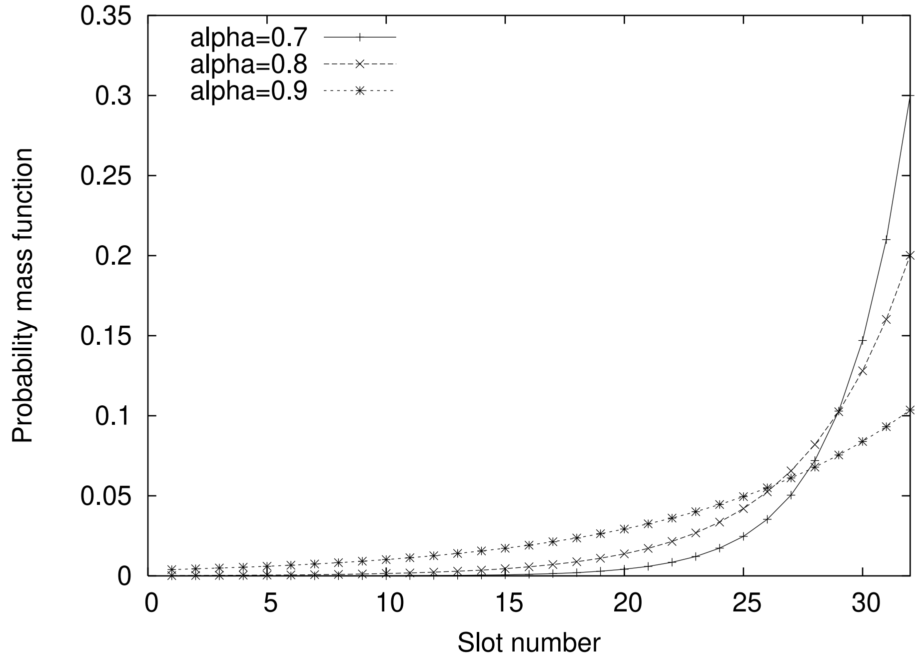

Figure 1: The probability distribution for the contention slot number that each Sift station chooses. We show various values of α, the parameter of the distribution.

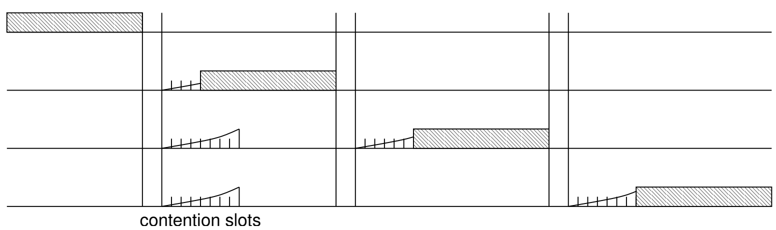

Figure 2: A timeline of four nodes running the Sift protocol. Shaded bars indicate packet transmission times. Every time the medium becomes idle, stations backoff at random according to the Sift distribution before transmitting. In this and Figure 4, stations add a constant time interval t_{difs} before backing off.

and there is a **collision** if more than one sensor 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; i.e. r is the first non-silent slot and there is no collision there. Finally, there is **success** if some sensor wins some slot in $[1, CW]$.

Sift uses a 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 distribution parameter. In this range of α, p_r increases exponentially with r, so the later slots have higher probability.

To motivate this choice, view each station'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 node chooses slot 1, that is an indication that N_1 is an overestimate of N, so each node updates its belief of the population

[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

size by decreasing N_1 to N_2, and proceeds to choose slot 2 with a certain probability in stage 2. If slot 2 is also silent, this belief is reduced to N_3 in stage 3, and so on; in general, N_r is the updated belief after there is silence in slots $1, \dots, r-1$.

Since N may in fact be any value from 1 to N_1, we want to keep the probability of success constantly high for each stage in our decision process. Specifically, we want the following properties to hold.

(i) The probability of success is high when $N = N_1$.

(ii) The probability of success is constant from one stage to another as the N-estimate N_r is reduced from N_1 to 1.

To get Property (ii), we now examine how a node calculates p_r in the rth stage of our imaginary decision process. Let p'_r be the probability[3] of choosing slot r, given that there is silence for slots $1, \dots, r-1$. Now, given silence before slot r, the probability of success in slot r is the probability that exactly one node picks slot r and is given by

$$N_r p'_r (1 - p'_r)^{N_r - 1} \approx N_r p'_r e^{-N_r p'_r} \tag{2}$$

for large N_r and small p'_r. Therefore Property (ii) holds if the term $N_r p'_r$ is approximately constant (note that it appears twice in Equation 2), so that the probability of success $N_r p'_r e^{-N_r p'_r}$ changes slowly as time progresses to the last slot.

We emphasize that we introduced N_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. Similarly, p'_r is merely a mathematical reinterpretation of the distribution p_r.

Since it is critical that Sift perform well for all $N < N_1$, we start with some scheme for determining $N_2, N_3, \dots, N_{CW}$, and find a distribution that gives us a constant $N_r p'_r$. To cover a large number of possible N values in a small number of slots CW, we choose an exponential scheme by which the belief of the active population size reduces, i.e.

$$\frac{N_{r+1}}{N_r} = \beta, \tag{3}$$

where β is a constant and $0 < \beta < 1$. With these $\{N_i\}$, we claim that if $\alpha = \beta$ then the distribution in Equation 1 satisfies the requirement that $N_{r+1} p'_{r+1} \approx N_r p'_r$.

Keeping in mind that each sensor picks its slot independently, and the competition for these slots ends once the first sensor succeeds, we have for sensor $\mathcal{S}$:

$$\begin{aligned} p'_r &= \Pr(\mathcal{S} \text{ chooses } r | \text{silence in earlier slots}) \\ &= \Pr(\mathcal{S} \text{ chooses } r | \mathcal{S} \text{ did not choose earlier slots}) \\ &= \frac{\Pr(\mathcal{S} \text{ chooses } r)}{\Pr(\mathcal{S} \text{ did not choose earlier slots})} \\ &= \frac{p_r}{1 - (p_1 + p_2 + \cdots + p_{r-1})} \\ &= \frac{(1-\alpha)\alpha^{(CW-r)}}{1 - \alpha^{(CW-r+1)}}, \end{aligned} \tag{4}$$

so

$$\frac{p'_r}{p'_{r+1}} = \frac{1 - \alpha^{(CW-r)}}{1 - \alpha^{(CW-r+1)}} \cdot \alpha \approx \alpha \tag{5}$$

example 802.11 limited the maximum contention window size to 1024 for commodity hardware at the time this paper was written. We will show later that above this population size, Sift's performance degrades gracefully.

[3] Although any individual p'_i is a probability, the set p'_r is not a probability distribution, since $\sum_{r=1}^{CW} p'_r > 1$.

for small α^{CW-r}.

If we set $\alpha = \beta$, then Equations 3 and 5 imply that $N_{r+1}p'_{r+1} \approx N_r p'_r$, thus giving Property (ii) above.

As for Property (i), we want to choose α so that the probability of success is high if $N = N_1$. Equation 4 implies that $p'_{CW} = 1$, so if all slots before CW are silent, then a node must choose the last slot. Therefore, α should be chosen so that a node in stage CW always believes it is the only active sensor, i.e. $N_{CW} = 1$. It follows from Equation 3 and $\alpha = \beta$ that $1 = N_{CW} = \alpha^{CW-1}N_1$, so $\alpha = N_1^{-\frac{1}{CW-1}}$.

Figure 3 plots the results of an experiment in which N sensors choose slots using the distribution in Equation 1, with $\alpha = 512^{-\frac{1}{31}} \approx 0.82$. Each point in the graph is for one run with N sensors (ignore the line above the points for now). Note that although we engineered our Sift probability distribution for a maximum number of sensors $N_1 = 512$, our 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 station, resulting in a collision. We ran the same simulation with α equal to 0.7, 0.9, and various other values. This verified that we had chosen the correct α, and that over the range $[0.7, 0.9]$, the success rate is not sensitive to the choice of α.

The graph 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.

2.2 Optimality results for contention window-based protocols

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 other information provided to it during the contention period.[4] In Appendix A, we prove the following theorem about the distribution p^* that optimizes the probability of success.

This theorem is significant because if the size of the colliding RTS packets (or small data packets, if RTS/CTS is turned off) is greater than the contention window size, then p^* is the distribution that yields the best latency acheivable by contention window-based protocols when $R = 1$.

Theorem 2.1. *Define $\pi(N)$ = Pr(success) when there are N contenders, and let*

$$f_1(N) = 0 \quad \text{and} \quad f_s(N) = \left(\frac{N-1}{N - f_{s-1}(N)}\right)^{N-1} \tag{6}$$

for $s = 2, \ldots, CW$. Then the maximum value for $\pi(N)$ is $f_{CW}(N)$, achieved with the distribution p^* where

$$p^*_r = \frac{1 - f_{CW-r}(N)}{N - f_{CW-r}(N)}(1 - p^*_1 - p^*_2 - \cdots - p^*_{r-1}) \tag{7}$$

for $r = 1, \ldots, CW-1$, and $p^*_{CW} = 1 - p^*_1 - \cdots - p^*_{CW-1}$.

Figure 3 shows the success probability of the Sift distribution as well as the theoretical success probability of the optimal distribution (shown as the line above the points). 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 often the case that N is unknown and hard to predict. This makes the optimal distribution impractical.

[4]These conditions exclude non-contention-window-based protocols like tree-splitting contention resolution. We address such protocols in Sections 2.4 and 4.1.

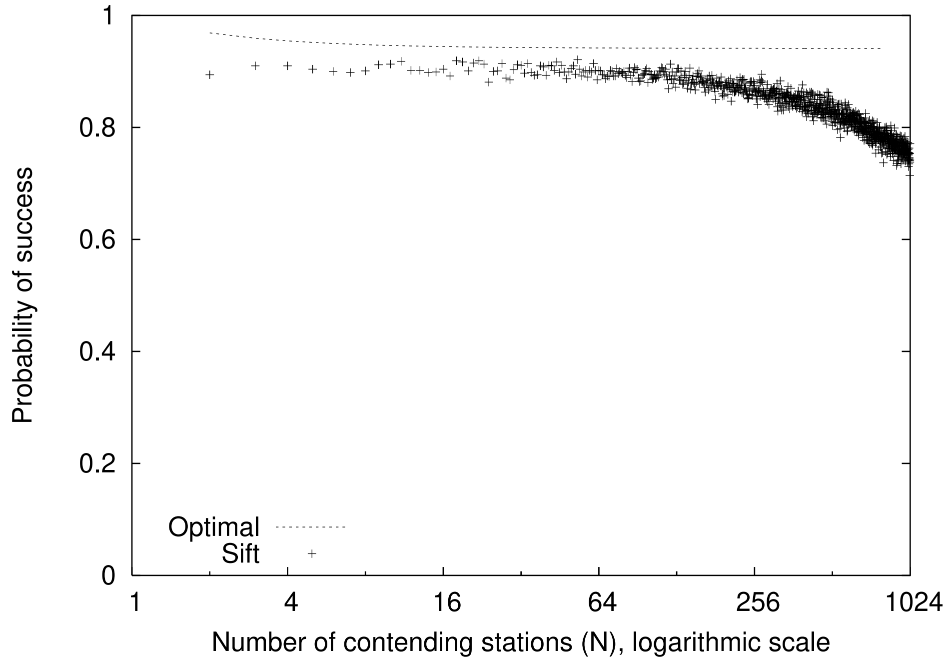

Figure 3: A comparison between Sift with $\alpha = 0.82$ and $CW = 32$, and an optimal protocol, with $CW = 32$. The optimal protocol has full knowledge of N; Sift has no knowledge of N. The Sift distribution shown above was engineered for a maximum value of $N = 512$ nodes, but degrades gracefully when N exceeds that figure.

2.3 Protocol specification

The pseudocode for the different states and state transitions in Sift is shown below.

```
Idle state

wait (channel idle)
if (recv frame for self)
   moveto Receive
end if
if (xmit queue not empty)
   moveto Contend
end if

Contend state

slot ←— pickslot()
wait t_difs + slot × t_slot
   if (channel busy)
      moveto Idle
   end if
Transmit frame
moveto AckWait

AckWait state

wait t_ACKTimeout
if (recv an ACK for self)
   discard frame
   moveto Idle
end if
Retransmit frame
moveto AckWait

Receive state

Check frame CRC
wait t_sifs
Send ACK
moveto Idle
```

The function `pickslot` picks a slot in $[1, CW]$ at random using the Sift distribution described in Equation 1. The directive **moveto** *state* exits the current block of code and starts executing the code in the new *state*. The directive **wait** waits for the specified time interval, or for the condition to become true. t_{slot}, t_{sifs}, t_{difs}, and $t_{\mathsf{ACKtimeout}}$ are all as defined in IEEE 802.11 [1]: t_{slot} is the minimum time separation such that if two stations transmit more than t_{slot} seconds apart, they will hear the onset of each others' transmissions. t_{difs} is a deferral time added to the beginning of a new data transmission. t_{sifs} is the deferral time added to the beginning of a data acknowledgment packet to allow a previously transmitting radio to turn around to receive an acknowledgment. Since $t_{\mathsf{difs}} = t_{\mathsf{sifs}} + 2 \times t_{\mathsf{slot}}$, acknowledgment packets from ongoing transmissions take precedence over fresh data packets. $t_{\mathsf{ACKtimeout}}$ is the total time to send the data and complete the receipt of an acknowledgment packet. Note that the timescale for each contention window slots is tens of microseconds: the slots are for contention purposes and do not carry data.

2.3.1 RTS/CTS

This state machine in Section 2.3 does not detail the initial RTS/CTS exchange, which almost eliminates collisions between very large packets. For large packet sizes, Sift uses the RTS/CTS exchange in almost the same way as IEEE 802.11. Instead of using the Sift backoff distribution to compete on data packets, we use it to compete on sending the RTS packet. We thus replace "frame" by "RTS" and "ACK" by "CTS" in the pseudocode above. Additionally, before sending a CTS a station must check that no other station has reserved the channel using an RTS. The reservation time is specified in the RTS.

In most of Section 3 we evaluate the performance of the protocol with RTS/CTS disabled, sending short data packets. In Section 3.3.4 we run a Sift experiment with RTS/CTS enabled, sending large data packets.

2.3.2 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. There are many possible ways to implement this suppression. In some scenarios, such as the last hop to a base station, this suppression is trivial to implement. Sensors can listen for R acknowledgement packets from the base station. In general, sensors can listen for R events timestamped within some time interval away from the event of interest. We plan to detail this process in future work.

2.4 Exploring the CSMA design space

Binary exponential backoff (BEB) is used in multiple-access networks with collision detection, such as the Ethernet [22]. Transmissions are "backed-off" to a random slot over a uniform contention window. When a transmission is unsuccessful due to a collision with another transmitting station, the uniform contention window is doubled in length. The resulting probability distribution for the contention slot that each station will choose will have the same probability mass spread over twice as many slots.

Bharghavan *et al.* proposed MACAW [3], a derivative of BEB, for wireless networks. Without some way to share information about the state of the wireless medium, BEB over wireless suffers from the same capture problem as Ethernet. A station that has just been successful in transmitting will reset its contention window to the minimal value, and is thus more likely to be the winner of subsequent contention competitions. The solution to this belongs to a class of techniques that we term *shared learning*. To improve the fairness of MACAW, stations *copy* the contention window of the winner in a contention competition. Another shared learning technique [4], also proposed by Bharghavan, is to increase the contention window if a collision is detected (as a result of bad packet CRC). The essence of the shared learning technique is that the knowledge gained by a station when it competes can be shared.

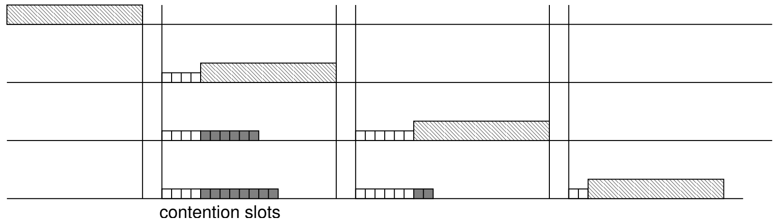

Figure 4: A timeline of four nodes running the 802.11 protocol. Large shaded bars indicate packet transmission times. The uniform backoff function is shown as small bars, with the remaining time left shaded dark gray if the medium becomes busy before the backoff countdown expires.

One might think that shared learning could help the problem of high rate of change of N with respect to time. The 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. However, this intuition is not true. 802.11 with shared learning, which we term *802.11/copy*, still suffers when N increases quickly. We substantiate this claim in Section 3.

Instead of shared learning, 802.11 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. 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 behaves differently from MACAW or BEB. Instead of picking a new random value in the backoff window, it *resumes* its countdown timer. Figure 4 shows the 802.11 protocol backing off in a simple network of four nodes.

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.

Garcés and Garcia-Luna-Aceves proposed a variant [13] of the FAMA family of protocols [11, 12] called CARMA (collision avoidance and resolution multiple-access). Like 802.11, CARMA uses carrier sensing during its backoff, but also uses a tree-splitting algorithm [2] to resolve collisions. On a population size of N, tree-splitting protocols like CARMA undergo about $\log N$ collisions before the first successful packet transmission, and $N - 1$ collisions in total to transmit N packets, a significant improvement on contention window-based protocols. Each time the belief decreases in Sift, we pay a time penalty of one contention slot in length. Each time the contention-resolution tree splits in CARMA, we pay a time penalty equal to the length of the colliding packet. This is significant because contention slots take less time than colliding RTS packets. For example, contention slots take 20 microseconds in 802.11, independent of bandwidth. RTS collisions take 320 microseconds in 802.11 at 1 MBps, and longer at lower bandwidths.

Tree-splitting algorithms also rely on stations waiting to transmit to be able to determine whether transmissions to other nodes were successful or not. Since interference is a property of the receiver, this is impossible to do with certainty, and hard to do in a noisy channel. In this regard, Sift is more practical than a tree-splitting algorithm, because Sift does not rely on being able to detect if others' receptions were successful.

Table 1 summarizes the design parameters that we have reviewed. From the table, it is clear that Sift explores a parameter of the contention window-based design space that has not been explored before. Section 3 shows that this particular choice in the design space results in good performance with respect to throughput, latency, and fairness.

3 Experiments

We ran experiments using version 2.1b9 of the *ns-2* [21] network simulator. Except where noted, the packet size in our simulations is 40 bytes, and all nodes are within range of a common base station. In our experiments, we compare Sift, IEEE 802.11, and 802.11/copy. All experimental results average 20 runs using different random seeds

Protocol	Contention Window	Shared learning	Memory
BEB	Variable	No	No
MACAW	Variable	Yes	No
802.11	Variable	No	Yes
802.11/copy	Variable	Yes	Yes
Sift	**Fixed**	**No**	**No**

Table 1: Some design parameters in the contention window-based CSMA space. The schemes with a variable-sized contention window use a uniform distribution to pick a slot. Sift requires neither shared learning nor memory to perform well.

Symbol	Meaning
N	Number of nodes responding to each event
R	Number of reports needed by the base station
d_{report}	Range within which sensors detect events

Table 2: Summary of the parameters that vary in our event-driven model and their meanings.

for each run, except the fairness experiments in Section 3.3.4.

3.1 Trace-driven workload

Constant-bit-rate or TCP flows do not always 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 want to obtain a trace of events reflective of what a real sensor network might see. We therefore propose an trace-driven workload to evaluate our design. In response to a simulated event near it, a sensor sends a small-sized report packet. If it hears R acknowledgements from the base-station, it suppresses its report, removing the packet from its transmit queue.

We modeled a sensor network deployed over an area of flat terrain. In our model, N nodes are scattered over an area to track the locations of people and cars in that area. If a sensor is less than d_{report} meters away from a moving object, it sends a 40 byte report packet reporting the presence of that object. The sensors in this model might be imagined as magnetometers, detecting the presence of metal from cars driving by, or vibration or shock detectors.

Rather than deploying this sensor network, we acquired video from the street scene shown in Figure 5. We detect movement using the motion detector software *Motion* [26] on the live image, which we modified to log motion events to a database. Each location event records the time of the motion and the x and y coordinates of the motion in the picture. We use this trace to drive most of our experiments.

To run an experiment with this trace-driven workload, we create an *ns-2* scenario where N sensors are placed uniformly at random on a two-dimensional plane. At the time given by an event in the database, all sensors within d_{report} meters of the location of that event send a 40 byte report packet. When there is continuous motion in the scene, events arrive at nearby sensors at the rate of 30 times per second. When there is no motion, no sensor senses any event. Note that the data used in our simulations are not the video images themselves, but rather event information from an analysis of motion in the video images.

Figure 6 shows the traffic pattern generated. Note that with this data, we not only capture the process modeling inter-arrival times of motion in a real-world setting, but we also capture small differences in density of sensor deployment,

Figure 5: An image captured from a video camera, showing an overheard view of a street in which vehicles and people move about. We use data from this stream to generate a realistic workload for sensor networks.

since the density of our randomly-placed sensors is not completely uniform.

3.2 Latency experiments

In these experiments, a constantly-changing subset of N nodes send event reports to a base station. We measure the time to receive one event report ($R = 1$), varying the number of sensors N that report the event. This experiment is independent of the workload we choose, provided that the workload does not saturate the wireless channel. As shown in Figure 7, when N is low, the minimum 802.11 contention window size is large enough to quickly resolve contention between the nodes. As the number of nodes reporting the event grows, however, the 802.11 contention window needs to grow before even one event report can be successfully sent. Using shared learning does not help the situation much because many stations broadcast underestimates of the optimal CW, resulting in a similar situation to plain 802.11. Sift does not need any time to adapt to large N, and so performs well over a large range of N. This graph shows the 7× latency reduction alluded to in the Abstract.

3.2.1 Desynchronization experiments

The above experiments were run when all nodes that were able to sense an event sensed it at exactly the same time. In reality, we expect that when a pair of nodes sense an event, their observations will be shifted in time by an amount on the order of a few milliseconds, due to propagation delays in the environment, variation of sensor electronics, and software system delays on the sensor nodes themselves. Consequently, the time at which the sensors attempt to report the event will also be desynchronized. The skeptic might suggest simply introducing a uniformly-chosen, random delay to the time a sensor chooses to send an event, claiming that this would reduce latency. We show that this is in fact not the case. Figure 8 shows the delay in delivering 64 reports of an event as a function of the variation in event sensing time. The traffic model is our constant-rate event model. Note that the performance of each protocol

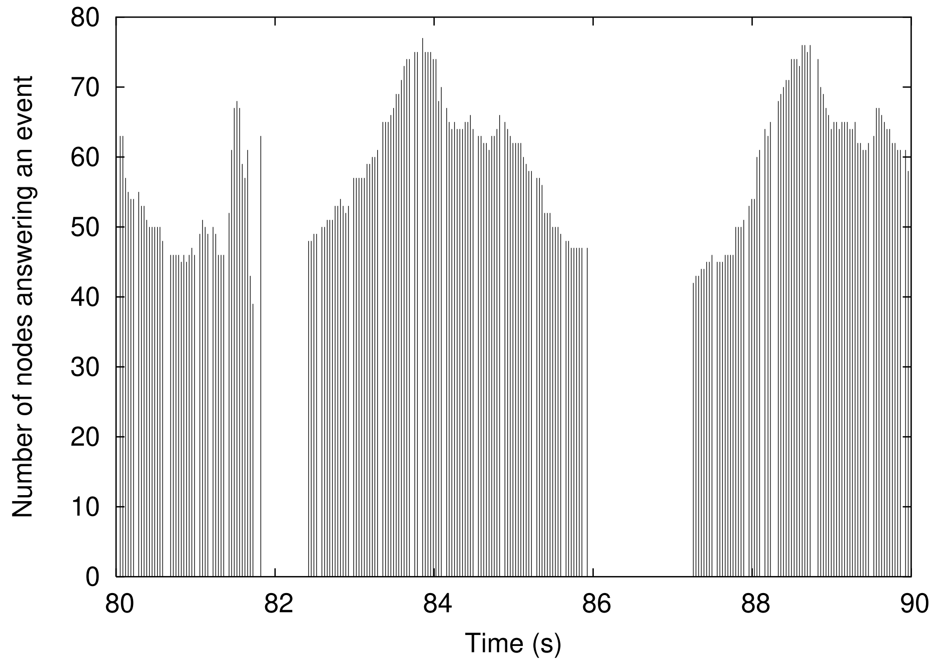

Figure 6: The number of nodes answering events in a trace-driven simulation using real-world event times acquired from a video stream using motion-detection software. We plot the number of imaginary sensors close enough to respond to the motion detected in the video frame as a function of the time of the motion. For detail, we show an excerpt of 10 seconds from a trace that lasted minutes.

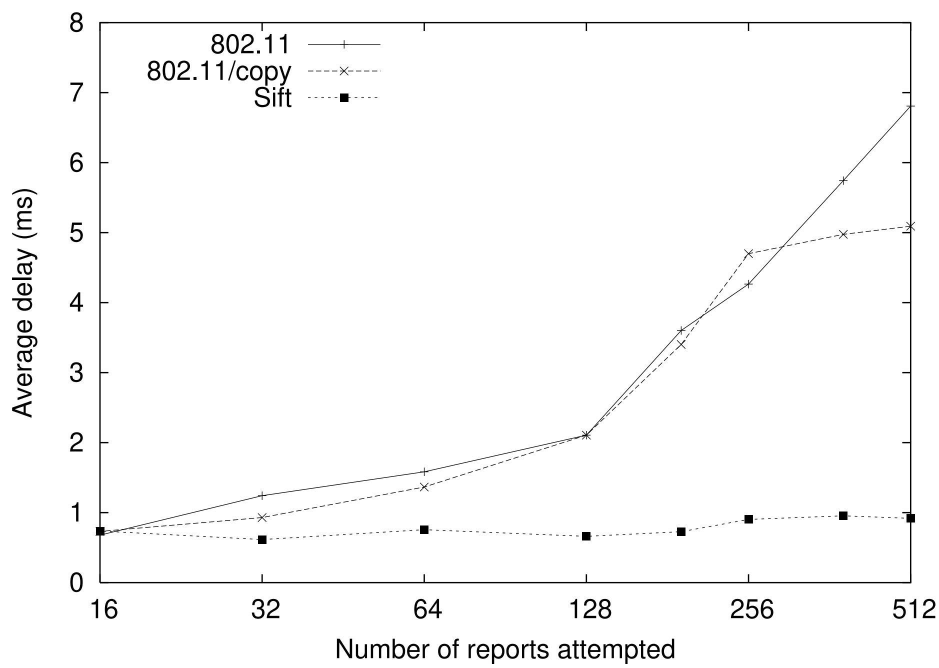

Figure 7: Average delay as a function of number of sensors activated to report an event. One report is required ($R = 1$), and all sensors detect the event at the same time. As the number of sensors reporting an event grows, window-based contention protocols require more time to adapt to the active population size.

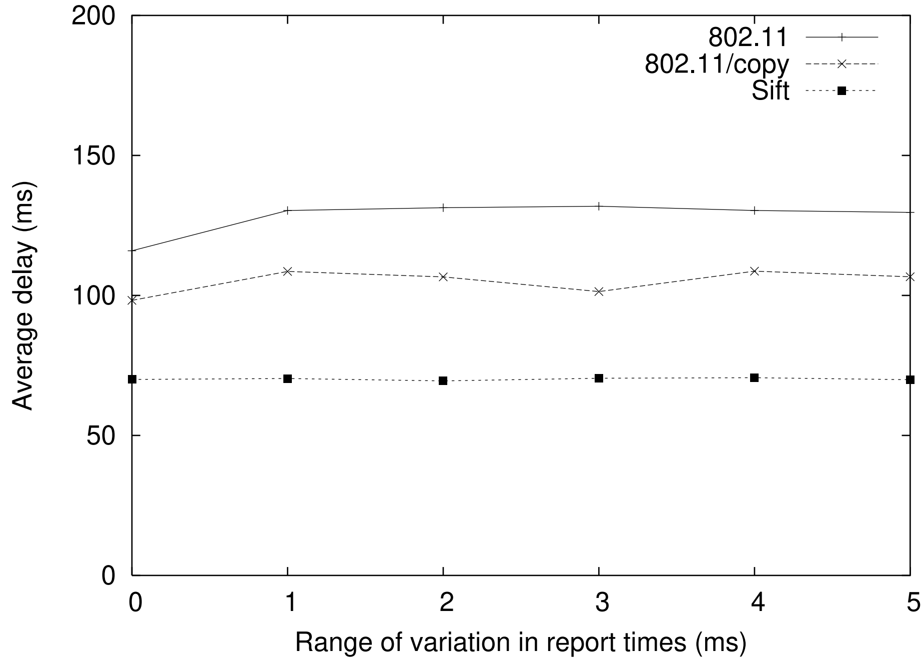

Figure 8: Average delay as a function of the maximum variation of the time that each sensor reports the event. Sensors report the event at the time of the event plus a random delay uniformly chosen between zero and the x-axis value. The number of sensor reports is 128 ($N = 128$) and 64 reports are required ($R = 64$). This experiment uses a constant-event-rate workload.

does not change as a function of variation in event sensing time. This means that unintentionally *or* intentionally adding variation to the event sensing times does not improve latency.

3.3 Throughput Experiments

We now compare the throughput achieved by Sift, 802.11, and 802.11/copy under a variety of workloads where $R = N$, or R is a significant fraction of N.

3.3.1 Constant-bit-rate traffic

In this experiment, we measure Sift's raw, steady-state throughput on a non-event-driven workload. The purpose of these experiments is to show that even though Sift performs extremely well under an event-based workload, it does not sacrifice much steady-state throughput in an ad-hoc network setting where there are some number of almost constant-bit-rate flows operating concurrently. Two, 8, and 32 CBR flows compete to send as much data as possible, using Sift, 802.11, and 802.11 with shared learning.

When the number of competing CBR flows is extremely small (two sources, the left-most graph in Figure 9), we see where Sift loses throughput compared to 802.11. This loss in throughput happens not because CBR flows incur collisions with Sift, but because the winning station wins in a late slot in the case of Sift. The winning 802.11 flow, however, wins in an earlier slot. Sift thus incurs several contention backoff slot delays per transmission, unlike 802.11. Note however, that a workload consisting of two CBR flows is not our target design. When the number of CBR flows increases even slightly, to eight, Sift performs almost as well as 802.11 (Figure 9, center). When the number increases further, Sift actually outperforms 802.11 in terms of raw throughput (Figure 9, left). This is

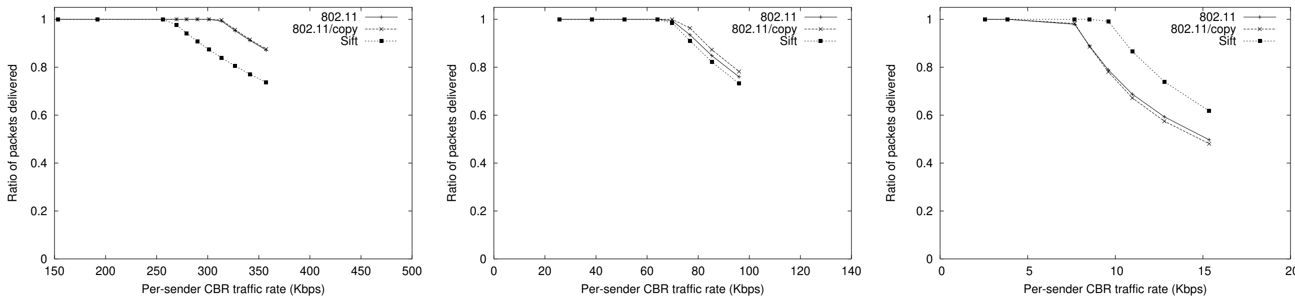

Figure 9: Packet delivery ratio as a function of per-sender CBR traffic rate. Left: 2 traffic sources; center: 8 traffic sources; right: 32 traffic sources.

because Sift does not incur many collisions, and when N increases, the shape of the Sift distribution makes the winning slot number decrease. This results in less wasted aggregate bandwidth.

3.3.2 Constant-rate events

Now we measure the time it takes to receive R events when the number of nodes reporting the event is fixed at 128. This experiment measures throughput of the network not in the steady-state, but in a dynamic situation where all nodes have just started to send and the network is still adapting to the sudden change. We see in Figure 10 that Sift achieves better throughput than 802.11 in this scenario. The reason for this is that Sift does not have to track the sudden change in the number of contenders ($\frac{dN}{dt}$), because the Sift distribution works well for a large range of N.

In Figure 11, we explore the Sift performance space when we vary both N and R. Consider first the bottommost curve for $R = 1$. The fact that it is constant with respect to N (the x-axis), means that no matter how many stations report an event, Sift can deliver one message with a small, constant latency. Now consider each additional curve in Figure 11. The distance from each curve to the curve below it measures the additional time needed to send more event reports. The fact that every curve in Figure 11 is constant with respect to N means that Sift can deliver R messages with a small, constant latency. This figure shows that Sift scales well both with respect to R and with respect to N for individual events.

3.3.3 Trace-driven events

Now, with the more demanding trace-based workload, we examine scaling the number of events R that Sift has to report. Since the traffic pattern is bursty and high-rate, when the number of reports needed 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 as a function of R. The results of this experiment are shown in Figure 12. Note the position of the knee of the various curves. Sift can continue delivering low-latency events for higher values of R because it has higher capacity under this workload.

3.3.4 Fairness experiments

Since Sift is a protocol for wireless nodes to access a shared medium, a natural question to ask is whether Sift allocates bandwidth to nodes fairly. It has been shown that 802.11 does not, but that minor changes to 802.11 can

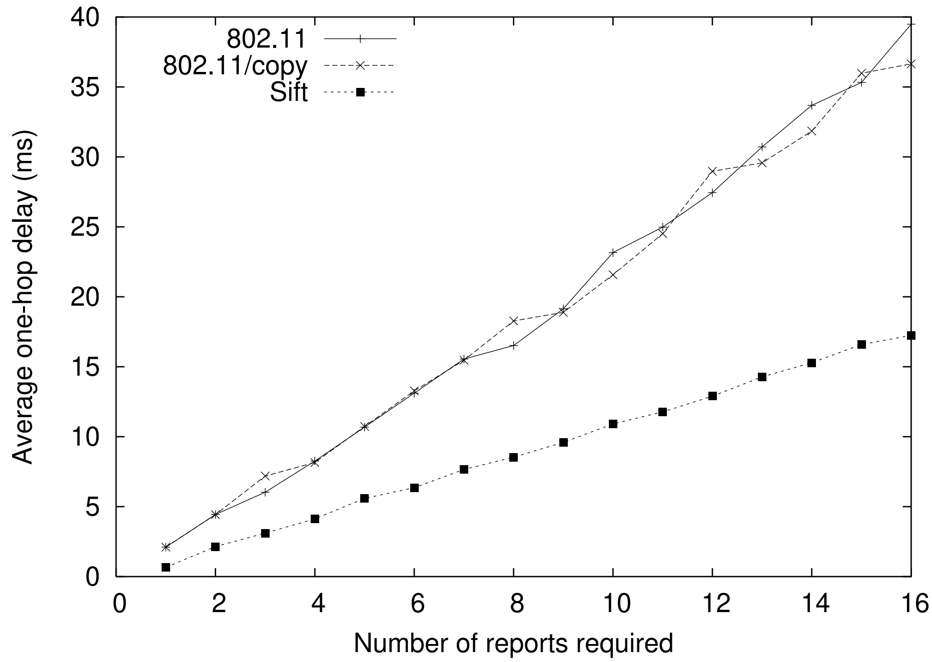

Figure 10: Average delay as a function of the number of event reports required when 128 sensors report an event ($N = 128$). All sensors detect the event at the same time. This experiment uses a constant-event-rate workload.

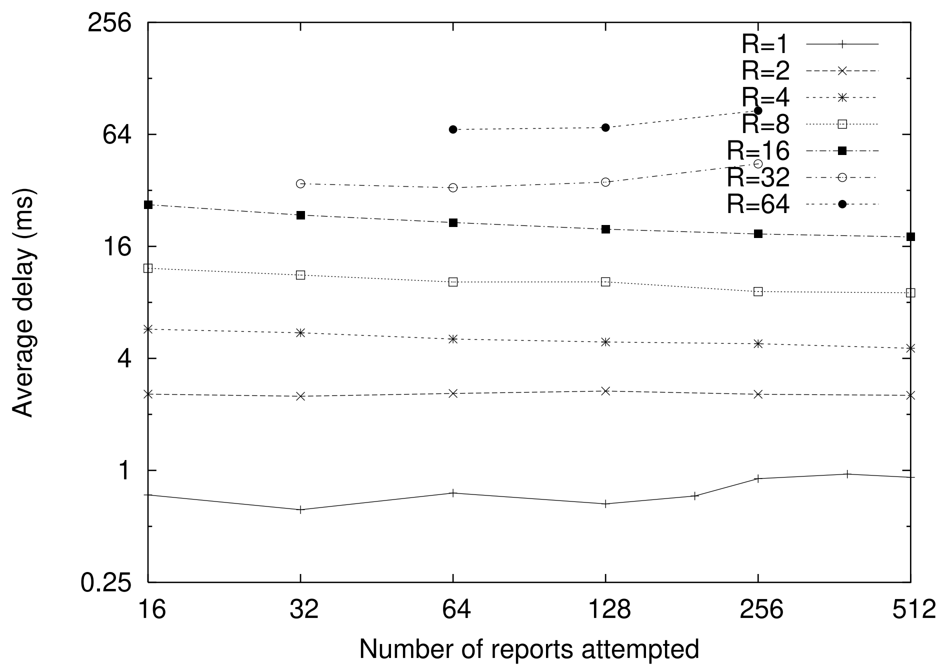

Figure 11: Average delay of Sift event reports as a function of N. We show curves for R (number of reports required) equal to powers of two between zero and 64. This experiment uses a constant-event-rate workload.

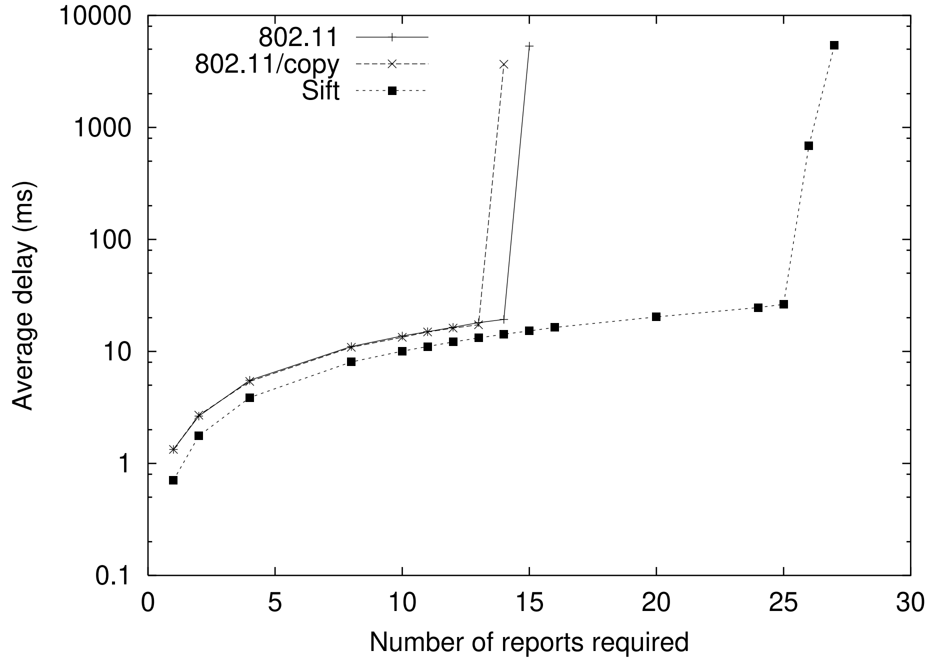

Figure 12: Average delay as a function of the number of reports R needed for each camera motion event. The sensor range in this experiment, d_{report} is fixed at 20. This experiment uses our trace-driven workload.

yield an extremely fair protocol [25].

We duplicate the experimental setup given by the authors of the distributed fair scheduling protocol (DFS) [25], to evaluate Sift. 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. We assume each node is backlogged, so the offered load is much more than the available wireless capacity.

Figure 13 shows the throughput achieved by each node in six seconds as a function of the cardinal node identifier. 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 if fairness becomes an issue.

4 Related work

In this section we discuss related MAC protocols. These fall into the categories of contention-based protocols, time-division protocols, and energy-conserving protocols. Finally, we discuss the relationship between Sift and the problem of reliable multicast feedback suppression.

4.1 CSMA

We compared Sift to 802.11, MACAW, and CARMA in Section 2.4. We now review more CSMA-related research.

Woo and Culler [28] compared the performance of various contention window-based MAC schemes, varying carrier sense time, backoff increase/decrease policies, and transmission deferral policies. All of their protocols used contention windows with the uniform distribution. Their evaluation of these conventional protocols was exhaustive, and

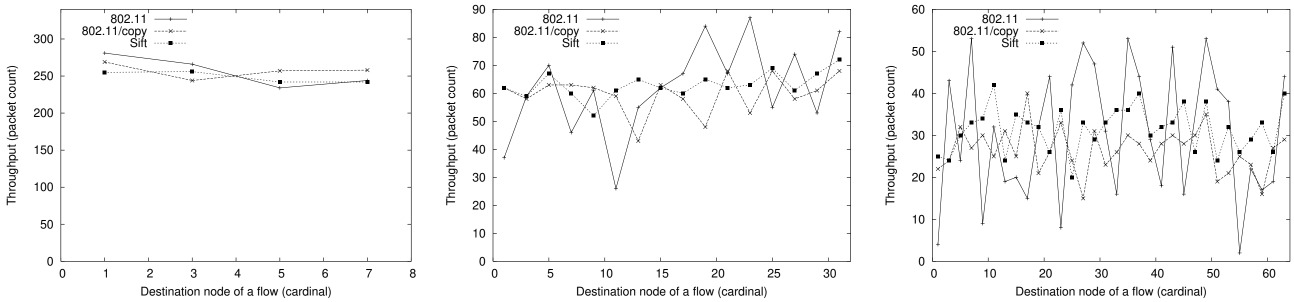

Figure 13: Fairness comparison of 802.11 and Sift. Left: eight nodes; center: 32 nodes; right: 64 nodes. In each experiment there are half as many flows as there are nodes. This experiment uses a CBR workload.

leads us to our present design point. They also noted that 802.11 is energy-inefficient, since it listens throughout its backoff period, and its backoff period can grow to be quite long. Sift compares favorably in this regard, because we use a fixed-size backoff window.

Calì, Conti, and Gregori proposed replacing the uniform-distribution contention window of 802.11 with a p-persistent backoff protocol [6, 7]. By estimating the population size, they chose p to maximize system throughput when *all* nodes *always* had a packet ready for transmission. They showed that 802.11 yields suboptimal throughput under this workload, and that their algorithm could approach optimal throughput under the same conditions. In contrast, we do not assume that all nodes always have a packet to send. We also do not rely on an estimator for N, which may be slow to adapt when N changes quickly.

4.2 Time-Division Multiple Access (TDMA)

For our problem, simple round-robin TDMA is highly suboptimal. A high rate of change of N (the subset of active nodes) with respect to time results in many unused time slots.

Time-spread multiple-access (TSMA) protocols assign each node a unique code that deterministically specifies the time slots in which the node has the right to transmit. In this way TSMA protocols try to combine the topology-independence of contention-based protocols with the deterministic bounds on message latency that TDMA offers. GRAND [9] is a TSMA algorithm where performance depends on the maximum node degree of the network D_{max}. In a highly-variable or completely-connected network, D_{max} could be $O(N)$ where N is the number of nodes in the network. This would result in a $O\left(N^2\right)$ frame length, which wastes large numbers of TDMA slots.

T-TSMA [10] runs a number TSMA protocols simultaneously, using round-robin TDMA to switch between protocols. Each protocol is tuned for nodes with neighborhood sizes equal to increasing powers of two. While T-TSMA scales in terms of asymptotic quantities, any threaded protocol must waste slots on running many protocols (albeit an asymptotically-small number). For example, when $N = 128$, T-TSMA must thread seven TSMA protocols, resulting an a $7\times$ latency penalty for some workloads.

4.3 Protocols for saving energy

S-MAC [32] 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 [15] 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. LEACH aims to extend the system lifetime of a sensor network. PAMAS [24] 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. Still other proposals for energy saving target higher layers, such as the transport layer in Kravets and Krishnan's proposal [18], for example. These protocols can be implemented completely independently of the Sift MAC.

4.3.1 Topology-control protocols

There have also been a number of proposals [8, 27, 29] for topology-control in wireless networks. The goal of these proposals is to save energy in multi-hop wireless networks either by electing some subset of the nodes in the network *coordinators*, or by reducing the radio ranges of nodes. The coordinator set or radio ranges are chosen in such a way that the network still remains connected. These protocols run above the MAC layer.

Although their goal is energy savings, if topology formation protocols could be adapted to take into account R in their decision procedure, it might be possible to provide an alternate solution to our problem. This is a non-trivial proposal, and we do not explore this option in the present work. We note that Sift can be used as a building block in the underlying MAC when these protocols run, to arbitrate access between the large numbers of nodes that need to rendezvous at the same time and elect coordinators.

4.4 Reliable multicast feedback

Slotting and damping is an idea useful in many contexts, including multicast group membership [5] and reliable multicast where feedback suppression is important for scalability. In reliable multicast, a receiver of a multicast packet sends feedback to the sender after a random delay, suppressing its feedback message if it hears a feedback message from another receiver. Nonnenmacher and Biersack proposed a non-uniform probability distribution similar to Sift's for scaling reliable multicast while avoiding feedback implosion [23]. However, unlike in our problem, the idea in multicast is to ensure that exactly one participant send a message quickly and all the others pick timer values as far as possible from this one participant to reduce the chance of duplicates.

5 Conclusion

This paper presented Sift, a MAC protocol for wireless sensor networks that performs well when spatially-correlated contention occurs and adapts well to changes in the active population size. Sift is tuned 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 an increasing, non-uniform probability distribution within a fixed-size contention window, rather than varying the contention window size as in many traditional MAC protocols. Our simulation results show that Sift improves over 802.11 in terms of report latency by up to a factor of 7 as the number of nodes reporting an event scales up to 512.

Figure 14 schematically summarizes the range of N and R where we showed that Sift has better performance than variable-window protocols like 802.11. Sift outperforms 802.11 both when the ratio R/N is low (in the bottom-left corner of Figure 14), and when both N and R are large (in the upper-right corner of the figure). To explain Sift's high performance, we also showed that Sift's success probability in each contention slot is only a little bit lower than that of a provably-optimal contention window-based MAC protocol that knows the number of contending nodes, N.

Sift is a simple window-based CSMA protocol that is easy to implement. We therefore believe that it can form an important lower-layer building block for a variety of sensor information dissemination protocols. These dissemination protocols can use the Sift MAC layer as a filter, dynamically sending control information informing nodes of a

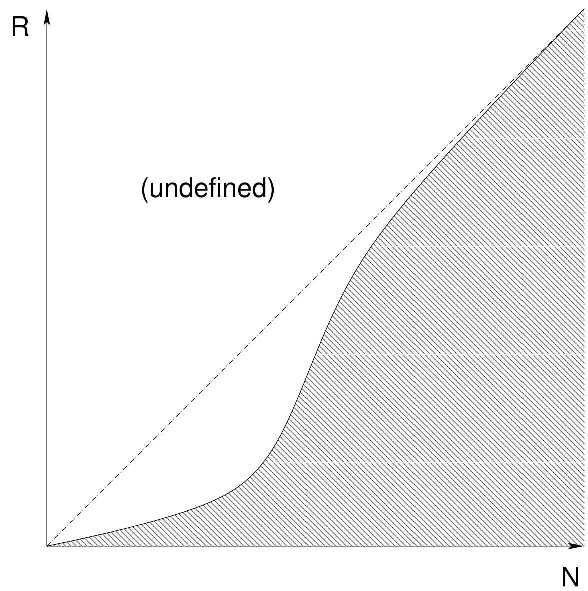

Figure 14: Phase plot showing regions of the N-R phase space (shaded) where Sift performs well in comparison to 802.11. Both protocols degrade when N increases to infinity. N represents the number of sensors that respond to each event and R is the number of event reports required for each event.

suitable value for R to use and improving report latency and network utilization. An interesting direction for future work is to integrate Sift with topology formation protocols like Span or GAF, to improve energy efficiency.

References

[1] Wireless LAN Medium Access Control and Physical Layer Specifications, Aug. 1999. IEEE 802.11 Standard.

[2] BERTSEKAS, D., AND GALLAGER, R. *Data Networks*, Second ed. Prentice-Hall, 1987.

[3] BHARGHAVAN, V. MACAW: A Media Access Protocol for Wireless LANs. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)* (London, UK, 1994), pp. 212–225.

[4] BHARGHAVAN, V. Performance Evaluation of Algorithms for Wireless Medium Access. In *Proceedings of the IEEE Computer Performance and Dependability Symposium (IPDS)* (Durham, NC, July 1998), pp. 86–95.

[5] CAIN, B., DEERING, S., KOUVELAS, I., FENNER, B., AND THYAGARAJAN, A. *Internet Group Management Protocol, Version 3*. IETF, Oct. 2002. RFC 3376.

[6] CALÌ, F., CONTI, M., AND GREGORI, E. IEEE 802.11 Wireless LAN: Capacity Analysis and Protocol Enhancement. In *Proceedings of the Conference on Computer Communications (IEEE INFOCOM)* (San Francisco, CA, 1998), vol. 1, pp. 142–149.

[7] CALÌ, F., CONTI, M., AND GREGORI, E. Dynamic Tuning of the IEEE 802.11 Protocol to Achieve a Theoretical Performance Limit. *IEEE/ACM Transactions on Networking 8*, 6 (December 2000), 785–799.

[8] CHEN, B., JAMIESON, K., BALAKRISHNAN, H., AND MORRIS, R. Span: An Energy-Efficient Coordination Algorithm for Topology Maintainence in Ad Hoc Wireless Networks. In *Proceedings of the Seventh International ACM Conference on Mobile Computing and Networking (MOBICOM)* (Rome, Italy, July 2001), pp. 85–96.

[9] CHLAMTAC, I., AND FARAGÓ, A. Making Transmission Schedules Immune to Topology Changes in Multihop Packet Radio Networks. *IEEE/ACM Transactions on Networking 2*, 1 (April 1994), 23–29.

[10] Chlamtac, I., Faragó, A., and Zhang, H. Time-Spread Multiple-Access (TSMA) Protocols for Multihop Mobile Radio Networks. *IEEE/ACM Transactions on Networking 5*, 6 (December 1997), 804–812.

[11] Fullmer, C., and Garcia-Luna-Aceves, J. Floor Acquisition Multiple-Access (FAMA) for Packet Radio Networks. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)* (Cambridge, MA, 1995), pp. 262–273.

[12] Fullmer, C., and Garcia-Luna-Aceves, J. J. FAMA-PJ: a Channel Access Protocol for Wireless LANs. In *Proceedings of the First International ACM Conference on Mobile Computing and Networking (MOBICOM)* (Berkeley, CA, 1995), pp. 76–85.

[13] Garcés, R., and Garcia-Luna-Aceves, J. J. Floor Acquisition Multiple Access with Collision Resolution. In *Proceedings of the Second International ACM Conference on Mobile Computing and Networking (MOBICOM)* (Rye, NY, 1996), pp. 187–197.

[14] Heidemann, J., Silva, F., Intanagonwiwat, C., Govindan, R., Estrin, D., and Ganesan, D. Building Efficient Wireless Sensor Networks with Low-Level Naming. In *Proceedings of the Symposium on Operating Systems Principles* (Alberta, Canada, October 2001), pp. 146–159.

[15] Heinzelman, W., Chandrakasan, A., and Balakrishnan, H. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS)* (January 2000).

[16] Intanagonwiwat, C., Govindan, R., and Estrin, D. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proceedings of the Sixth International ACM Conference on Mobile Computing and Networking (MOBICOM)* (Boston, MA, 2000), pp. 56–67.

[17] Kahn, J., Katz, R., and Pister, K. Mobile Networking for Smart Dust. In *Proceedings of the Fifth International ACM Conference on Mobile Computing and Networking (MOBICOM)* (Seattle, WA, 1999).

[18] Kravets, R., and Krishnan, P. Power Management Techniques for Mobile Communication. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)* (Dallas, TX, 1998), pp. 157–168.

[19] Madden, S., Franklin, M., Hellerstein, J., and Hong, W. TAG: a Tiny AGregation Service for Ad-Hoc Sensor Networks. In *Proceedings of the Fifth USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (Boston, MA, December 2002).

[20] Madden, S., Szewczyk, R., Franklin, M., and Culler, D. Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks. In *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications* (Callicoon, NY, June 2002).

[21] McCanne, S., and Floyd, S. ns Notes and Documentation. `http://www.isi.edu/vint/nsnam/`.

[22] Metcalfe, R., and Boggs, D. Ethernet: Distributed Packet Switching for Local Computer Networks. *Communications of the ACM 19*, 7 (July 1976), 395–404.

[23] Nonnenmacher, J., and Biersack, E. Optimal Multicast Feedback. In *Proceedings of the IEEE Computer and Communications Societies (INFOCOM)* (San Francisco, CA, 1998), vol. 3, pp. 964–971.

[24] Singh, S., Woo, M., and Raghavendra, C. S. Power-Aware Routing in Mobile Ad Hoc Networks. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)* (Dallas, TX, 1998), pp. 181–190.

[25] Vaidya, N., Bahl, V., and Gupta, S. Distributed Fair Scheduling in a Wireless LAN. In *Proceedings of the Sixth International ACM Conference on Mobile Computing and Networking (MOBICOM)* (Boston, MA, 2000), pp. 167–178.

[26] Vreeken, J. The *Motion* Software Motion Detector. `http://motion.sourceforge.net`.

[27] Wattenhofer, R., Li, L., Bahl, V., and Wang, Y.-M. Distributed Topology Control for Wireless Multihop Ad-hoc Networks. In *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)* (Anchorage, AK, April 2001), pp. 1370–1379.

[28] Woo, A., and Culler, D. A Transmission Control Scheme for Media Access in Sensor Networks. In *Proceedings of the Seventh International ACM Conference on Mobile Computing and Networking (MOBICOM)* (Rome, Italy, 2001).

[29] Xu, Y., Heidemann, J., and Estrin, D. Geography-Informed Energy Conservation for Ad Hoc Routing. In *Proceedings of the Seventh International ACM Conference on Mobile Computing and Networking (MOBICOM)* (Rome, Italy), pp. 70–84.

[30] Ye, F., Lu, S., and Zhang, L. GRAdient Broadcast: A Robust, Long-Lived Large Sensor Network. Tech. rep., UCLA, 1999.

[31] Ye, F., Luo, H., Cheng, J., Lu, S., and Zhang, L. A Two-Tier Data Dissemination Model for Large-Scale Wireless Sensor Networks. In *Proceedings of the Eighth International ACM Conference on Mobile Computing and Networking (MOBICOM)* (Atlanta, GA, 2002).

[32] Ye, W., Heidemann, J., and Estrin, D. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)* (New York, NY, June 2002), pp. 1567–1576.

A Proof of optimality

We present here the proof for Theorem 2.1 and a corollary that provides some intuition for the recursive function f_s that defines the optimal distribution p^*. To reduce clutter, let $K = CW$. For convenience, we begin by restating the theorem. Recall that $\pi(N)$ = Pr(some sensor wins some slot) when there are N sensors; this is nontrivial only for $N \geq 2$. Define

$$f_1(N) = 0 \quad \text{and} \quad f_s(N) = \Big(\frac{N-1}{N - f_{s-1}(N)}\Big)^{N-1} \tag{6}$$

for $s = 2, \ldots, K$. Then the distribution p^* that maximizes $\pi(N)$ is given by

$$p_r^* = \frac{1 - f_{K-r}(N)}{N - f_{K-r}(N)}(1 - p_1^* - p_2^* - \cdots - p_{r-1}^*) \tag{7}$$

for $r = 1, \ldots, K-1$.

Proof

$$\begin{aligned} \pi(N) &= Np_1^*(1-p_1^*)^{N-1} + Np_2^*(1-p_1^*-p_2^*)^{N-1} \\ &\quad + \cdots + Np_{K-1}^*(1-p_1^* - \cdots - p_{K-1}^*)^{N-1} \\ &= N\sum_{s=1}^{K-1} p_s^*\Big(1 - \sum_{r=1}^{s} p_r^*\Big)^{N-1}. \end{aligned}$$

(Since $N \geq 2$, if all sensors choose slot K, then there is a collision.) If the maximum occurs in an interior point ($0 < p_s^* < 1$ for all s), we must necessarily have

$$\frac{\partial}{\partial p_j^*}\Big(\frac{\pi(N)}{N}\Big) = 0 \quad \text{for} \quad j = 1, \ldots, K-1$$

(p_K^* is determined by $p_K^* = 1 - p_1^* - \cdots - p_{K-1}^*$), i.e.

$$(N-1)\sum_{s=j}^{K-1} p_s^*\Big(1 - \sum_{r=1}^{s} p_r^*\Big)^{N-2} = \Big(1 - \sum_{r=1}^{j} p_r^*\Big)^{N-1}. \tag{8}$$

We prove by induction on $i = 1, \ldots, K-1$ that

$$(N - f_i(N))p^*_{K-i} = (1 - f_i(N))\Big(1 - \sum_{r=1}^{K-(i+1)} p^*_r\Big). \tag{9}$$

For $i = 1$, we set $j = K - 1$ in Equation 8 to get

$$(N-1)p^*_{K-1}\Big(1 - \sum_{r=1}^{K-1} p^*_r\Big)^{N-2} = \Big(1 - \sum_{r=1}^{K-1} p^*_r\Big)^{N-1},$$

so $Np^*_{K-1} = 1 - \sum_{r=1}^{K-2} p^*_r$; i.e. Equation 9 is true for $i = 1$, since $f_1(N) = 0$.
Assume now that Equation 9 is true for $i = \ell$, so

$$(N - f_\ell(N))p^*_{K-\ell} = (1 - f_\ell(N))\Big(1 - \sum_{r=1}^{K-(\ell+1)} p^*_r\Big) \tag{10}$$

$$\text{and} \quad (N-1)p^*_{K-\ell} = (1 - f_\ell(N))\Big(1 - \sum_{r=1}^{K-\ell} p^*_r\Big) \tag{11}$$

Setting $j = K - (\ell + 1)$ in Equation 8 gives

$$\begin{aligned}&(N-1)p^*_{K-(\ell+1)}\Big(1 - \sum_{r=1}^{K-(\ell+1)} p^*_r\Big)^{N-2}\\ &+(N-1)\sum_{s=K-\ell}^{K-1} p^*_s\big(1 - \sum_{r=1}^{s} p^*_r\big)^{N-2} = \big(1 - \sum_{r=1}^{K-(\ell+1)} p^*_r\big)^{N-1},\end{aligned}$$

so, by Equation 8,

$$\begin{aligned}&(N-1)p^*_{K-(\ell+1)}\Big(1 - \sum_{r=1}^{K-(\ell+1)} p^*_r\Big)^{N-2}\\ &+\Big(1 - \sum_{r=1}^{K-\ell} p^*_r\Big)^{N-1} = \Big(1 - \sum_{r=1}^{K-(\ell+1)} p^*_r\Big)^{N-1}.\end{aligned}$$

It follows from Equation 10 and Equation 11 that

$$\begin{aligned}&(N-1)p^*_{K-(\ell+1)}\Big(\frac{N - f_\ell(N)}{1 - f_\ell(N)}\Big)^{N-2}\\ &+\Big(\frac{N-1}{1 - f_\ell(N)}\Big)^{N-1} p^*_{K-\ell} = \Big(\frac{N - f_\ell(N)}{1 - f_\ell(N)}\Big)^{N-1} p^*_{K-\ell}\end{aligned}$$

and, by Equation 6,

$$\begin{aligned}(N-1)p^*_{K-(\ell+1)} \quad &+ \quad f_{\ell+1}(N)\frac{N - f_\ell(N)}{1 - f_\ell(N)}p^*_{K-\ell}\\ &= \frac{N - f_\ell(N)}{1 - f_\ell(N)}p^*_{K-\ell}.\end{aligned}$$

This and Equation 10 imply

$$(N-1)p^*_{K-(\ell+1)} = (1 - f_{\ell+1}(N))\Big(1 - \sum_{r=1}^{K-(\ell+1)} p^*_r\Big);$$

equivalently,

$$(N - f_{\ell+1}(N))p^*_{K-(\ell+1)} = (1 - f_{\ell+1}(N))\Big(1 - \sum_{r=1}^{K-(\ell+2)} p^*_r\Big),$$

i.e. Equation 9 is true for $i = \ell + 1$, completing the induction. We get Equation 7 from Equation 9 by setting $i = K - r$.

One can further verify (with more calculus) that Equation 7 defines a maximum. □

We next provide some intuition for $f_s(N)$, and show that the maximum value for $\pi(N)$ is $f_K(N)$.

Corollary Suppose all sensors use p^* with the same N value. Let $p^{*\prime}_r$ = Pr(sensor $\mathcal{S}$ chooses slot r | $\mathcal{S}$ does not choose any slot before r). Then

(i) $p^{*\prime}_r = \frac{1-f_{K-r}(N)}{N-f_{K-r}(N)}$ for $r = 1, \ldots, K-1$.

(ii) $f_{K-r}(N) = (1 - p^{*\prime}_{r+1})^{N-1}$ for $r = 0, \ldots, K-2$.

(iii) $f_{K-r}(N) = \Pr(\text{there is a winner} \mid \text{no sensor chooses any slot before } r+1)$ for $r = 0, \ldots, K-1$.

(iv) The maximum value of $\pi(N)$ is $f_K(N)$.

Proof

(i) This follows from $p^{*\prime}_r = \frac{p_r}{1-p_1-\cdots-p_{r-1}}$ and Equation 7.

(ii) By the definition of $f_s(N)$ and (i),

$$f_{K-r}(N) = \Big(\frac{N-1}{N - f_{K-r-1}(N)}\Big)^{N-1} = (1 - p^{*\prime}_{r+1})^{N-1}.$$

(iii) We do an induction on $r = K-1, K-2, \ldots, 1, 0$. For $r = K-1$, if no sensor chooses any slot before $r + 1 (= K)$, then all sensors collide ($N \geq 2$) in slot K, so there is no winner; in this case, (iii) is true since $f_1(N) = 0$.

Assume (iii) is true for $r = \ell$. For $r = \ell - 1$, if no sensor chooses any slot before $r + 1 (= \ell)$, then there is a winner if and only if the winner chooses slot ℓ, or no sensor chooses slot ℓ and the winner picks a later slot. Hence
$\Pr(\text{there is a winner} \mid \text{no one chooses any slot before } \ell)$

$$\begin{aligned} &= Np^{*\prime}_\ell(1-p^{*\prime}_\ell)^{N-1} + (1-p^{*\prime}_\ell)^N f_{K-\ell}(N) \\ &\qquad \text{by the induction hypothesis} \\ &= (1-p^{*\prime}_\ell)^{N-1} \quad \text{by (i)}. \end{aligned}$$

By (ii), this is $f_{K-(\ell-1)}(N)$, thus completing the induction for (iii).

(iv) $\pi(N) = \Pr(\text{there is a winner}) = f_K(N)$ by (iii). □