RMAC: a randomized adaptive medium access control algorithm for sensor networks

Suprakash Datta
Computer Science and Engineering Department
York University, Toronto, ON, Canada
Email: datta@cs.yorku.ca

***Abstract*— It is believed that in the near future, there will be a large-scale deployment of wireless sensor networks for sensing, collecting and transferring information about the environments the sensors are placed in. Due to their peculiar characteristics and goals, sensor networks differ sharply from traditional wired and wireless networks, and so many problems have to be solved afresh for such networks. In this paper, we study the medium access control problem and focus on a class of Time-Division-Multiple-Access (TDMA) protocols called demand assignment protocols. We propose a simple, adaptive algorithm RMAC for heterogeneous wireless sensor networks. RMAC improves on existing demand assignment algorithms like RMAV [1], DQRUMA [2] and the protocol proposed in [3] in several ways. First, it uses Maximum-Likelihood estimation to reduce packet delays and uses a simple prediction scheme to adapt to changing traffic patterns. Second, in most demand assignment algorithms, a transmitted packet reserves a timeslot for the next packet in the same message. Thus, when multi-packet messages are being transmitted, a new message may have have to wait for a long time before it can be transmitted; i.e., the algorithms are unfair to new sessions. Our algorithm provides a parameter that controls this fairness property of the algorithm. We present simulation results to compare the performance of our algorithm with RMAV [1], DQRUMA [2] and the protocol proposed in [3].**

I. Introduction

It is believed that the near future will see a large-scale deployment of wireless sensor networks for sensing, gathering and disseminating information about the environment in which the networks exist. Sensor networks are large-scale networks built from small, cheap, untethered low-power devices that are placed in an *ad hoc* manner. The nodes communicate with each other using wireless messages, self-organize and cooperate in performing sensing and information collection and dissemination tasks. The scale, power restrictions and *ad hoc* nature of these networks makes them rather different from traditional wired and even wireless networks and so many problems have to be solved afresh for sensor networks. There is already a significant amount of existing work in this area. Until now, most research has focused on homogeneous networks built with low-power devices. However, recent technological advances have made commercially available several types of sensors (see, e.g., [4]). The smaller, lower power devices can be deployed on a large scale and the larger, significantly higher power nodes can be used to function as the "backbone" of a heterogeneous sensor network.

The focus of this paper is designing a medium access control (henceforth referred to as MAC) algorithm for heterogeneous sensor networks. There is a potentially large number of low-power sensor nodes that may communicate with a larger node. In reality, not all sensor nodes have packets to transmit at any given time. The larger node does not know how many nodes have packets to send. It may not even know the total number of nodes that are in its transmission range since nodes often run out of batteries and die. We wish to develop a simple, efficient protocol that can be used by the sensor nodes.

The problem has some similarities to the same problem in cellular networks where a large number of low-power mobile stations communicate with a high-power base station. However, there are significant differences. First, we assume that our sensor network is static – i.e., the nodes are not mobile. Second, the applications, and hence both the traffic and the service requirements are completely different. Third, although the nodes on the backbone network have more powerful batteries than the sensor nodes, they still have limited power and thus differ sharply from base stations in cellular networks. Finally, the small size of typical messages and the relatively predictable packet destinations in sensor networks make the problem simpler and less general than its counterpart in cellular networks.

We note that the problem being studied is a part of the larger problem of minimizing end-to-end delays in sensor networks. It is our belief that the backbone nodes are more suitable for deploying existing algorithms like S-MAC [5] or TRAMA [6], whereas our algorithm solves the problems unique to sensor nodes well. Thus, an effective solution to the larger problem may be obtained by combining our algorithm with existing algorithms.

A. Related work

We make no attempt to present a survey of literature on sensor networks. The reader is referred to the nice surveys [7]–[9] and to [10] for applications of sensor networks. In this subsection, we survey MAC protocols relevant to our work.

1) MAC protocols for wireless networks:: There is a very large number of papers and protocols on MAC protocols for wired and wireless networks. We refer the reader to [11] for results on wireless MAC protocols, to [12] for a survey of MAC protocols for mobile *ad hoc* protocols, to [13] for energy efficient protocols for wireless networks and to [14] for performance evaluation of MAC protocols for wireless ATM networks.

2) MAC protocols for sensor networks:: There are several MAC layer protocols designed specifically for sensor networks. The S-MAC protocol [5] performs the dual task of medium access control and self-organization. Nodes form virtual clusters and synchronize sleep schedules. They also use message passing to reduce contention. The T-MAC protocol [15] extends S-MAC using variable length duty cycles. Crudely speaking, nodes try to send data in bursts and sleep between bursts. Like S-MAC, it uses virtual clusters. DMAC [16] uses variable length duty cycles. The length of the cycles are changed adaptively to minimize latency. It has a simple prediction model for data arrival, and tries to reduce latency by adding offsets to the beginning of sleep periods of nodes depending on the position of the node in the tree, so that a multihop message does not have to wait at every node for the node to wake up. RAP [17] and the protocol proposed in [18] use orthogonal codes to perform Code Division Multiple Access (CDMA) to prevent collisions. Finally, TRAMA [6] is a collision-free protocol in which nodes exchange schedules and elect leaders in a distributed manner. In addition, TRAMA conserves battery power by allowing nodes to enter an idle state when they have no data to send.

3) Demand assignment MAC protocols: There are several characteristics of our problem that rule out many existing approaches. Since a backbone node does not know how many sensor wish to send packets, polling all nodes (a strategy used in some wired local area networks) not only wastes precious battery power at the backbone node but also wastes bandwidth and therefore adds considerable delay to the packets being transmitted. In addition, sensor nodes typically have a single wireless transmission channel on which all logical uplink and downlink channels have to be time-multiplexed. This suggests a time-division-multiplexing-access (TDMA) approach may be most suitable for this problem. Finally, a large number of low level sensor nodes connect to a larger backbone node forming a star-like subgraph. This implies that a centralized MAC algorithm can be run at the backbone node. All these observations point to the suitability of a demand assignment protocol [11] for our problem.

Demand assignment protocols were originally proposed for wireless ATM networks with Quality-of-Service (QoS) requirements. They allocate bandwidth to nodes to meet these requirements. Several demand assignment protocols have been proposed in the literature, including RMAV [1], DQRUMA [2], MASCARA [19], Sift [20] and the protocol proposed in [3]. All of these protocols operate in rounds and there are three phases in each round. The first phase is the contention or reservation round. In this round sources contend for reserving *minislots*, using a randomized algorithm. In the second phase, the backbone node announces the identities and the time slots "captured" by the nodes that succeeded in the contention phase. The third phase consists of the nodes listed in the second phase actually transmitting the packets in their *slots*. This 3-phase structure improves efficiency by preventing collisions in the third phase, and by using minislots that are much smaller than the slots. Typically, demand assignment protocols improve efficiency by allowing reservation requests to be *piggybacked* on packets transmitted in slots so that a node that transmits packets this round and has more packets to send does not have to compete for minislots again in the next round.

The protocols named before differ in the design of the contention phase. RMAV [1] is an adaptive algorithm in which there is one minislot in every round. Nodes contend by attempting to reserve this slot. If there is a collision, nodes back off for a random period of time governed by a probability p that is set adaptively by the base station. The algorithm used to adapt p is simple – if there is a collision in the single minislot, p is doubled; if the minislot is idle, p is halved. Otherwise, p remains unchanged.

In DQRUMA [2], nodes attempt to reserve transmission slots using either slotted ALOHA or a more complex algorithm called the binary stack algorithm. Since the performance of both are very similar, we only describe the variation using slotted ALOHA. In DQRUMA, there is one minislot as in RMAV, but empty slots are converted into contention rounds with M minislots each (M is typically a small fixed number)[1]. DQRUMA uses a *harmonic* backoff protocol, i.e., after k collisions, a node flips a coin with probability of head $\frac{1}{1+k}$ and tries to capture a minislot if the result of the toss is a HEAD.

In [3], the authors propose an algorithm in which nodes contend to reserve minislots using slotted ALOHA. Nodes which are successful get to transmit packets. The number of minislots are changed adaptively: if the number of minislots with collisions is larger than the number of empty minislots, then the number of minislots are doubled in the next round. Otherwise the number of minislots in the next round are halved.

It is worth pointing out that all these papers are designed to deal with multimedia traffic with variable size messages, multiple traffic classes and different Quality-of-Service (QoS) constraints.

Sift [20] is a contention-based TDMA protocol designed for sensor networks. It uses non-uniform probabilities of choosing minislots. Roughly speaking, a node makes an initial choice of a reservation minislot, but may change that choice later as it observes what happens in the minislots before the chosen one in the same round.

While all the protocols just described are adaptive, they suffer from some drawbacks. RMAV and DQRUMA both use backoff protocols, which have proved to be very hard to analyze even for wired networks. The protocol proposed by [3] does not use backoff, and they demonstrate that the algorithm performs better than RMAV, but the basis for their heuristic of comparing the number of minislots with collisions with the number of empty minislots is not clear from their paper. Sift was designed for sensor networks but it requires the sensors to remain awake during all minislots, monitor the network for

[1]The authors mention that the algorithm could make DQRUMA more adaptive in several ways but do not investigate this further in the paper.

transmissions and accordingly recompute their transmission probabilities in the next minislot.

Our proposed protocol, RMAC, has significant similarities with, and was inspired by the DQRUMA protocol, the RMAV protocol and the protocol proposed in [3]. We address the drawbacks of these protocols mentioned above. Unlike RMAV and DQRUMA, RMAC does not employ backoff mechanisms. Like all these three, RMAC uses a slotted ALOHA algorithm in the first phase. Like Aboelaze et al, we adjust the number of minislots adaptively. Unlike any of the three papers, we use a predictive model to compute the number of packets expected in a round.

B. Fairness Issues in MAC protocols

There have been several efforts at studying the issue of fairness and its impact on performance in networks. We do not attempt a detailed survey of results in this area in this paper, and mention a few relevant papers. Ozugur et al [21] propose interesting solutions for this problem for general wireless networks. In our opinion, they seem a little too complex for low power sensor nodes. Nandagopal et al [22] translate the fairness problem into a contention resolution problem and derive a backoff algorithm for achieving proportional fairness. Koksal et al [23] investigate short-term fairness of MAC protocols using several statistical measures.

C. Bayesian Estimation

We now list some results from Bayesian Estimation theory that are used in the paper. Given two dependent events A,B, Bayes Theory allows us to compute the *a posteriori* probabilities $\Pr(B = j|A)$ in terms of the conditional probabilities $\Pr(A|B = j)$ using the well-known formula:

$$\Pr(B = j|A) = \frac{\Pr(A|B = j)\Pr(B = j)}{\sum_{j=1}^{c} \Pr(A|B = j)\Pr(B = j)} \quad (1)$$

There are many possible estimators that one can use, each with its advantages. The reader is referred to the textbook [24] and the references therein for more details. We discuss three well-known estimators, the Maximum Likelihood Estimator (MLE), the Maximum *a posteriori* Probability (MAP) estimator and the Bayes estimator for minimizing mean squared error.

The MLE is given by the expression $\max_j \Pr(B = j|A)$. The MAP is given by the expression $\max_j \Pr(B = j|A)\Pr(A)$, i.e., it is the value of j that maximizes the *a posteriori* probabilities. The Bayesian estimator for minimizing the mean squared error is given by $\sum_{j=1,2,\ldots} j \cdot \Pr(B = j|A)$, i.e., it is the mean of the conditional distribution in Equation 1.

D. Our contributions

The contributions of this paper are as follows:

1) We separate the problem of adaptively varying the number of minislots into three parts. First, we need to estimate the number of packets in the current round. Second, we need to predict the number of packets in the next round. Finally, we need to compute the number of minislots in the next round. We use Maximum-Likelihood estimation to solve the first problem. We show that the heuristic used by RMAV [1] and by Aboelaze et al [3] can be explained using this idea. The prediction problem is solved by a simple linear predictor. The last subproblem is solved by optimizing the expected delay of a packet ignoring the queuing delay (i.e. the time a packet spends in the buffer before it gets a chance to attempt a minislot reservation is ignored in this calculation). All these ideas are used to design a randomized, adaptive MAC algorithm called RMAC.
2) We propose a very simple generalization to demand assignment protocols like RMAV, DQRUMA, and the algorithm by Aboelaze et al to improve their fairness properties. We use the same idea to modify RMAC and show the performance-fairness tradeoff with this modification.
3) We investigate via simulation studies the performance of RMAC. Our experiments show that RMAC has better performance than DQRUMA and the algorithm of Aboelaze et al. The latter was shown in [3] to perform better than RMAV.

E. Outline of this paper

This paper is organized as follows. Section II explains our model and lists the performance metrics we consider in this paper. Section III describes our MAC protocols. Section IV provides both analytical and experimental performance evaluation results of our protocols. Finally, Section VI presents some conclusions and future work.

II. Our model and metrics

A. Network Model

We consider a heterogeneous sensor network model where there are a large number of static, low-power nodes connecting to a backbone of larger, higher-power nodes. Our protocol is designed for communication between the higher and lower level nodes. We will sometimes refer to the lower level nodes as sensor nodes and the higher level nodes as *host* nodes. Since our protocol runs in a distributed manner, we will henceforth focus on one host node connected to N sensor nodes, and treat this subnetwork as the network being studied.

We assume that the MAC protocol is not required to perform the task of self organization, unlike, e.g., [5]. We expect that node placement has been done with the goal of forming the desired topology, but recognize that placement strategies are imprecise and there may be considerable variation from the planned placement. We assume also that the low-power nodes always connect to a single host node, and that this (static) assignment is made when the network self-organizes. We also assume that the host nodes themselves communicate with a more complex protocol than the one proposed in this paper, e.g., a collision-free protocol like the one described in [6]. This assumption is justified by the fact that host nodes have higher power, more data to send, and are less prone to failures than sensor nodes. Further, the network of host nodes can be

organized as a multi-hop ad hoc network whose underlying graph has low node-degree. The advantage of this assumption is that interference between neighboring hosts is reduced.

Since sensor networks can operate in unpredictable environments, there could be significant number of corrupted or lost packets. We assume that providing reliable transfers is not the task of the MAC protocol; such concerns must be handled by higher layers in the network protocol architecture.

B. Node Model

We assume that the sensor nodes have unique identifiers. It is our belief that this would be true for almost all applications which require sensor networks. It is crucial for our algorithm to have access to a good pseudo-random number generator. Without unique identifiers, ensuring distinct random number sequences among different nodes is a difficult problem. The unique identifiers can be used to seed the generator, and so it becomes very improbable that two sensor nodes use the same (pseudo-)random numbers.

Our algorithms are based on Time-Division-Multiple-Access (TDMA) and so we need reasonably good time synchronization between nodes. Since sensor nodes often have one or two communication channels, TDMA has often been used, and so this assumption has often been made in the literature (see, e.g., [15]). The clock synchronization required for our algorithm is local. As long as each of the sensor nodes have their clocks synchronized to their host node, the protocol works fine. Protocols that achieve this feasibly in sensor networks exist in the literature (e.g, the RBS protocol [25]).

Finally, we assume that battery-power conservation is the primary focus of sensor nodes, and therefore, all protocols running on them must allow nodes to sleep between periods of activity to conserve battery life.

C. Traffic model

We assume that the sensor network does not carry traffic with strict QoS requirements (e.g., real-time traffic) or different classes or with different priorities. Since sensor nodes typically exchange small amounts of information at a time, we assume that each message fits within a single packet. Often queries are made periodically and so the traffic in the network after the initial topology building consists of occasional "talk spurts" of constant bit rate traffic separated by silent periods. We model this behavior using three simple traffic models. The simple model (called "random traffic" in this paper) assumes that packets are generated by a Bernoulli process. At every timestep, a packet is generated at each sensor with probability λ/N and no packets are generated with probability $1-\lambda/N$, where N is the total number of sensor nodes and $0 < \lambda < 1$.

The second model (called "random bursty" model in this paper) generalizes random traffic in that bursts of size $B > 1$ are generated by a Bernoulli process. Thus we assume that at every timestep, a burst of B packets are generated at each sensor with probability λ/NB and no packets are generated with probability $1-\lambda/NB$, where N is the total number of sensor nodes and $0 < \lambda < 1$.

Finally, we consider a simple ON-OFF burst model in which the packet generator is a two-state discrete-time Markov Chain (the states being ON and OFF). In the ON state, a packet is generated every timestep. No packets are generated in the OFF state. At each timestep, the transition probability from ON to OFF is a and OFF to ON is b. It can be shown easily that the probability of being in the ON state is given by $b/(a+b)$ [26], and so we can choose a, b that satisfy $\frac{a}{a+b} = \frac{\lambda}{N}$.

D. Fairness

For many applications, it is desirable that a MAC algorithm be fair. For example, a network monitoring algorithm may wish to get the state of all the nodes at (approximately) the same time. Fairness often reduces efficiency. In demand assignment algorithms, the ability of a node to bypass the contention phase by piggybacking a reservation request with a transmitted packet decreases the expected delay. However, this also implies that a node that always has packets never has to contend for minislots, unlike nodes which are sometimes empty, and this implies that the former nodes may have lower expected delays than the latter. So the former nodes have an unfair advantage in demand assignment MAC algorithms. We address this issue in the next section (Section III).

The fairness of MAC algorithms is affected by another unexpected factor called the *capture effect* [27]. This effect happens when a collision between nodes A,B, is *not* detected since the signal from A is so much stronger than the signal from B that the receiver is able to decipher the message from A and effectively ignore the message from B. In such situations, the utilization of the medium increases at the cost of fairness – the receiver is able to convert a collision minislot into a successful one, but B will never be able to send a message whenever it collides with A. In this paper, we ignore the capture effect, and defer such studies for future work.

E. Performance Metrics

The most important performance metric for our purposes is the delay of a packet. Since we are concerned with MAC protocols, we define delay on a per-hop and not end-to-end basis. We do not try to capture short-term fairness in this paper. The measure of fairness used in this paper is the variance of the delay across nodes. More precisely, we consider the expected delays of packets sent by each node and study the variance of these quantities.

III. Our MAC protocol

As mentioned before, our protocol belongs to the general class of demand assignment algorithms. Like other demand assignment algorithms, the algorithm proceeds in rounds. As described in Section I-A.3, each round has three phases (see Figure 1). The broad structure of our algorithm (and many demand assignment algorithms) is as follows.

The first phase has a variable number of fixed-size minislots, decided and announced by the host. Each node that has a packet to transmit chooses a minislot uniformly at random, and transmits a reservation request in that minislot. A node is

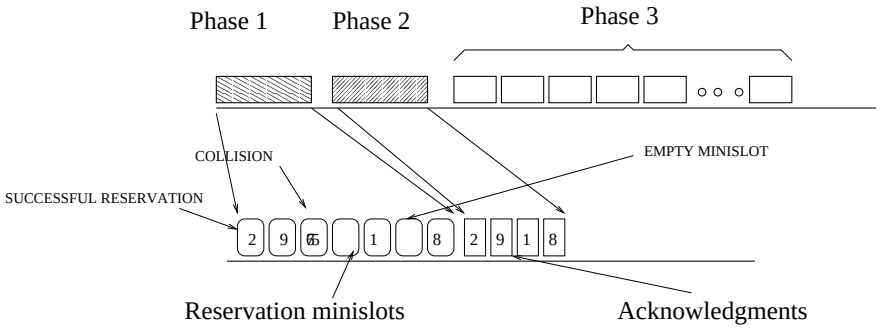

Fig. 1. The three phases of our protocol

also allowed to avoid the contention phase by piggybacking a request for a slot in the current round on a packet that it sent the previous round. The host reads the minislots and detects collisions. Then, in the second phase, it sends an acknowledgment packet, which announces the number and identities of nodes that have successfully reserved minislots. In the final phase, these successful nodes transmit their packets in their slots. Thus, there are no collisions in this phase. A source that is unsuccessful in reserving a minislot will try the same process again in the next round. Next, we make some observations about our algorithm.

1) **Batching requests:** For simplicity, we assume that packets which arrive midway through a round get "batched", i.e., they wait until the next round to take part in the minislot reservation process.
2) **Need for acknowledgments:** The explicit acknowledgment in the second phase is necessary for two reasons. Firstly, it solves the hidden terminal problem [28]. If all nodes could hear each other, then each node would know which nodes successfully reserved minislots. Secondly, it allows nodes to transmit their requests and sleep.
3) **Use of node identifiers:** Apart from the already mentioned use of node identifiers in generating distinct random numbers, the identifiers are necessary to address the *capture effect*; if we ignored occurrences of this effect then the acknowledgment could only list minislots and since each node knows the minislots it chose, there would be no collisions in the next phase.
   We note that although one could use the less significant bits of the sensor readings for random number generation, this could result in problems when sensor readings are heavily correlated.

A. Design issues

1) **Computing the number of minislots:** Unlike [2], we do not set backoff periods for sources after a collision. Unlike [1], and like [3], we adaptively vary the number of reservation minislots. This is a complicated problem for the following reasons.
   a) A host cannot observe the actual number of sensors competing for minislots. However, it knows the number of minislots it offered and also the number of minislots in which there are collisions, the number of minislots which are empty and the number of minislots that are reserved successfully.
   b) Even if the number of packets that will arrive in a certain round is known, computing the number of minislots that should be offered in that round is not obvious. Note that having too few minislots result in a small success rate in the contention phase, and using too many minislots increase the expected delay of packets.
   c) Even if the number of packets that arrived at the last round is known, the number of packets that will arrive in the current round is not known, and this is an issue in computing the number of minislots that should be used in this round.
2) **Fairness-efficiency tradeoff:** As mentioned before, algorithms that allow nodes to bypass the contention phase create unfairness in bandwidth allocation to nodes. This is true for our algorithm and also other algorithms that allow the same strategy, including DQRUMA and the algorithm in [3]. To our knowledge, a study of the efficiency-fairness tradeoff has not been carried out by relaxing or restricting the ability of nodes to bypass the contention phase.

B. Our solutions

We now outline our solutions to the design problems outlined in Section III-A. To our knowledge, this is the first decomposition to the problem of finding the number of minislots in a demand assignment MAC algorithm. We expect these ideas to be useful in improving existing algorithms like DQRUMA as well.

1) The estimation problem: We solve the problem of computing the number of nodes $m(t)$ that competed for minislots by posing it as a classical Bayes Estimation Theory problem: given the observed numbers of minislots $n_e(t)$ that are empty, the number $n_c(t)$ of minislots that have collisions and the number $n_s(t)$ of minislots have been reserved successfully, what is the most likely number of nodes $m(t)$ that could have resulted in these observations? Thus the problem is one of estimation of $B = m(t)$ given $A = \langle n_c(t), n_s(t), n_e(t) \rangle$.

The probabilities $\Pr(n_s, n_e, n_c|m, n)$ can be computed using simple results about the combinatorics of the classical balls-and-bins problem. We state the formula here and prove it in Lemma 2 in Appendix A.

$$\Pr(n_s = j, n_c = k, n_e = n - j - k|m, n) = \frac{1}{n^m}\binom{n}{j, k}\binom{m - j - k - 1}{k - 1} \quad (2)$$

In order to use the Bayes formula (1) for either MAP or Bayes estimation, we need the *a priori* probabilities $\Pr(B)$. Since the *a priori* distribution is not available, in this paper, we only deal with the MLE which does not require the *a priori* distribution.

a) Simplifying the MLE for our problem: In Lemma 4 in Section in the Appendix, we show that the Maximum Likelihood estimator for $m(t)$, $\text{MLE}[m(t)]$ can be simplified to obtain the following expression.

$$\widehat{m}(t) = \text{MLE}[m(t)] = n_s(t) + 2n_c(t).$$

This has several interesting implications.

1) We do not actually have to compute the expressions in Equation(2). Apart from consuming time, these computations require arithmetic with binomial coefficients and factorials which may be difficult for sensor networks.
2) The heuristic used by RMAV (respectively Aboelaze et al [3]) can be explained in terms of the MLE. Recall that their heuristics doubles the expected length of the backoff period (respectively the number of minislots) if $n_c(t) > n_e(t)$ and halves the backoff period (respectively the number of minislots) if $n_e(t) > n_c(t)$. Note that the actual number of minislots is $n(t) = n_c(t) + n_s(t) + n_e(t)$ and the estimated number of nodes competing for minislots is $\widehat{m}(t) = n_s(t)+2n_c(t)$. So the comparison $\widehat{m}(t) > n(t)$ is equivalent to the one used by RMAV and Aboelaze et al. Note that the comparison $\widehat{m}(t) > n(t)$ makes perfect intuitive sense: if the estimated number of nodes is greater than the number of minislots offered, the latter should be increased.

2) Optimal number of minislots: We compute the number of minislots $n(t)$ to be used in round t when the number of competing nodes $m(t)$ is known by obtaining an expression for the total delay of a packet in the first phase (ignoring the time the packet spent waiting for its turn in attempting a minislot reservation) and computing the number of minislots that optimize this expression. In this paper, we use the formula $n(t) = Cm(t)$, where C is a constant. We show in Section IV-B that the value of C obtained by optimizing the part of the delay described above is $C = 1$. In other words, the number of minislots should be equal to the number of nodes that compete for minislots in a round.

3) The prediction problem: Let us note first that predicting packet arrivals is of little use when packets are generated by a random process with an unknown mean. We have investigated some extensions of our algorithm that try to estimate the mean by observing the number of arrivals over a long time. We do not discuss those extensions for two reasons: first such extensions are of limited use unless we are sure that the inputs are generated by a stationary random process and second the algorithm can estimate indirectly but not observe the number of new arrivals in a timestep. In this paper, we focus on designing an adaptive algorithm and therefore include a simple predictor described below.

We use a linear model to solve the problem of predicting the number of packets that are expected in the current round, given the number of packets that arrived in the past k rounds, where k is a small constant. We use $k = 2$ in this paper, and compute the predicted number of packets in round $t + 1$, $p(t + 1)$ using the formula

$$p(t + 1) = \widehat{n}(t) + \alpha(\widehat{n}(t) - \widehat{n}(t - 1)).$$

Thus the predicted number of packets is the estimated number of packets in the last round plus α times the *gradient* of the number of packets in round t. While we make no claims of optimality of this simple predictor, we provide two justifications for its choice: first, it is very simple and thus appropriate for sensor networks, and second, it seemed to perform well in our simulations. The value of the parameter α was also chosen empirically.

4) The fairness problem: Since the unfairness in demand assignment algorithms arises from allowing a node to bypass the contention phase, we let the host reject such requests with some probability f. Note that $f = 1$ corresponds to the original demand assignment MAC algorithm, and $f = 0$ corresponds to piggybacked reservations not allowed by the algorithm. For $0 < f < 1$, we expect the fairness properties to improve and efficiency to degrade as f decreases.

This gives us a natural generalization of many existing MAC algorithms including RMAV [1], DQRUMA [2] and the protocol proposed in [3], and allows us to study the fairness-efficiency tradeoff as a function of f.

C. The RMAC Algorithm

Our algorithm puts together the ideas outlined in the last few subsections. We investigated several other variants of these ideas. For example, we tried using a smoothed estimate of $\widehat{m}(t)$ over the last few rounds, but the performance was inferior to the non-smoothed estimate. We also used more complicated prediction schemes but did not see any significant performance improvement. The steps of the algorithm in round t are summarized in Figure I. Note that this algorithm runs at each host node, and is the "server" side of the protocol. The "client" side, running at the sensor nodes, is extremely simple, and consists of contending for the advertised minislots and reading the ACKs sent by the host to determine the slot it should transmit in.

We observe that our algorithm is expected to work best for low to medium arrival rates. If λ is close to one, polling-based algorithms are close to optimal. Also, for $f = 1$, we expect our algorithm to win over DQRUMA and the algorithm in

RMAC(α, f)
1 **for** $round = 1, 2, \ldots$
2 **do** $n(t) = \widehat{m}(t-1) + \alpha(\widehat{m}(t-1) - \widehat{m}(t-2))$
3 Let the sensor nodes attempt to reserve minislots.
4 Record $n_c(t), n_e(t), n_s(t)$
5 $\widehat{m}(t) = n_s + 2n_c$
6 Send ACKs to all nodes that successfully reserved
7 minislots, and those who send piggybacked
8 reservation requests
9 Accept packets sent in slots, and also piggybacked
10 reservation requests
11 Accept each piggybacked request with probability f

TABLE I
THE RMAC ALGORITHM

[3] when λ is small. Otherwise, nodes are rarely empty and our adaptive adjustment of the minislots does not have a great impact on performance. In fact the aggressive increase of the number of minislots by the algorithm in [3] is appropriate for high arrival rates.

D. Advantages of RMAC

Our algorithm has several advantages.

1) **Simplicity:** RMAC is a simple protocol, and requires very little computational power other than a decent random number source.
2) **Energy conservation:** RMAC allows sensor and host nodes to sleep for extended periods. A sensor node need only be awake during its reservation attempt, the acknowledgment sent by the host and its slot if it was successful in reserving a minislot. The begin times of each phase is announced by the host. More importantly, the host can insert delays between phases and perhaps sleep if it so desires, without affecting the operation of the protocol.
3) **Fairness:** RMAC allows the designer to tradeoff efficiency for fairness using a very simple strategy.

IV. PERFORMANCE ANALYSIS

The main objective of this section is to compute the optimal value of the constant C used to determine the number of minislots. To simplify the analysis, we assume that the burst size $B = 1$.

First, we assume that the system reaches a steady state. When packets are generated by Bernoulli processes, it is straightforward to prove that this is true using the theory of Markov Chains [26]; the proof is omitted.

For analysis purposes, we view our protocol as having two distinct parts. The first part consists of the *queuing subsystem*. A packet enters this subsystem when it arrives at a node and waits until packets ahead of it have successfully reserved minislots and have been transmitted. A packet leaves this subsystem at the end of a round in which the packet in front of it has has been transmitted to the host. It then enters the *contention subsystem*. At steady-state, the expected total delay D of a packet is given by $D = D_c + D_q$, where D_c is the expected delay in the contention subsystem and D_q is the expected delay in the queuing subsystem.

A. Computation of D_c

The presence of piggybacked requests makes the derivation of D_c rather complicated. Therefore, we assume that such requests are never allowed, which correspond to the case $f = 0$ in our algorithm. While this assumption is invalid for high arrival rates, we expect our algorithm to work best for low arrival rates, and in such cases our analysis yields reasonably good results.

D_c is upper bounded by the product $E[L]E[R]$, where $E[R]$ is of the expected number of rounds a packet has to try until it reserves a minislot, and $E[L]$ is the expected length of a round. Note that R and L are not independent random variables as we have implicitly assumed above. Our expression is an upper bound since we have neglected the fact that packets may reach their destinations before the end of a round.

At steady-state, the expected number of packets coming in during round r is $E[m(r)] = \overline{m}$. This counts both packets attempting to reserve minislots for the first time and those trying after one or more failed attempts. Also, the expected number of minislots is $E[n(r)] = CE[m(r)] = C\overline{m}$.

Given $\overline{m}$ packets, $C\overline{m}$ minislots, the probability that a packet succeeds in reserving a minislot is $\frac{1}{e^{1/C}}$ and the expected number of successful reservations is $\frac{\overline{m}}{e^{1/C}}$ using Lemma 3. So, the steady-state throughput is $\overline{m}(1 - \frac{1}{e^{1/C}})$, and the expected number of rounds a packet requires to be successful is $E[r] = e^{1/C}$.

The expected length of a round is $E[L] = 2C\overline{m} + S\frac{1}{e^{1/C}}\overline{m}$, since there are $E[n] = C\overline{m}$ minislots and $\frac{\overline{m}}{e^{1/C}}$ slots, one for each successful reservation. Each slot is of length S and the acknowledgments for the reservation is assumed to take one minislot per successful node. Thus

$$\begin{aligned} D_c &= E[r]E[L] \\ &= e^{1/C}\left(C\overline{m} + (1+S)\frac{1}{e^{1/C}}\overline{m}\right) \\ &= \overline{m}\left(Ce^{1/C} + S + 1\right). \end{aligned}$$

B. The optimal value of C

We can now compute the value of C that optimizes D_c. Using elementary calculus, we find that the minimum value of D_c occurs at $C = 1$ and is given by $D_c = \overline{m}\,(e + S + 1)$. Using Little's Theorem [29], the maximum arrival rate $\lambda_{\max}$ that the contention subsystem can support satisfies $D_c = \frac{\text{number of packets in the subsystem}}{\lambda}$, since the expected number of packets in this subsystem is $\overline{m}$. This yields $\lambda_{\max} = \frac{1}{e+S+1}$, and the maximum arrival rate at any node is given by $\lambda_{\max}/N = \frac{1}{N(e+S+1)}$ where N is the number of nodes that can compete for minislots.

Note that the maximum throughput by a centralized optimal algorithm (that knows the status of all nodes) is $1/S$, since each packet takes time S to be transmitted. Our algorithm

is distributed and must therefore "waste" some time in coordination tasks. Our expression for $\lambda_{\max}$ also shows that the coordination overhead can be reduced by increasing S, i.e., having bigger slots. However, bigger slots are not feasible in real sensor networks, since they increase the lengths of the energy-draining transmission and reception periods for sensors, and therefore drain the battery faster.

We also note that the presence of piggybacked reservations allow the algorithm to improve its efficiency.

V. Simulation Results

Due to space limitations, we present a subset of our experimental results. We simulated several algorithms using a "homegrown" discrete-event simulator written in C. All time units in this section are normalized to the duration of a minislot. While this produces small numbers (note the maximum arrival rate in our network later in this paragraph), we believe that the duration of a minislot is determined by the sensor network technology and connectivity and is a fundamental parameter of a network.

In our simulations, a network of 100 sensor nodes were connected to a backbone node for 100,000 time units (i.e., minislot durations). We assumed that the duration of a slot is $S = 15$ (although we have experimented with other values of S, we do not present the results here, since they are essentially similar). The maximum arrival rate λ for $f = 0$ (according to our analytical results) satisfied $\lambda \leq \frac{1}{15+e+1} = 0.05342$. So the maximum per-node arrival rate for the simulations is $0.05342/100 = 0.0005342$. We simulated both $B = 1$ and $B > 1$. The results presented here are for $B = 4$, since the results are similar to those for higher B.

The experiments reported in this section are meant to answer the following questions.

- How does the expected delay vary with the parameter α used by the prediction algorithm?
- For a good value of α chosen above, how does our algorithm compare to existing algorithms like DQRUMA and the algorithm by Aboelaze et al, as well as to the ideal algorithm?
- What is the efficiency-fairness tradeoff as the fairness parameter f introduced in this paper is varied?

A. Effect of α on expected delay

As mentioned, we do not expect the prediction algorithm to significantly improve performance when random or bursty random traffic are being used. This is indeed borne out by our experiments.

The prediction algorithm works much better with the ON-OFF burst model. In Figure 2, we show the variation of the expected delay with α. For this experiment, we used $\lambda = 0.0002$, $a = 0.25$, and $N = 100$.

B. Comparison with other algorithms

In this section, we compare the performance of RMAC with the Ideal (optimal centralized) algorithm, DQRUMA, and the algorithm in [3] for random traffic.

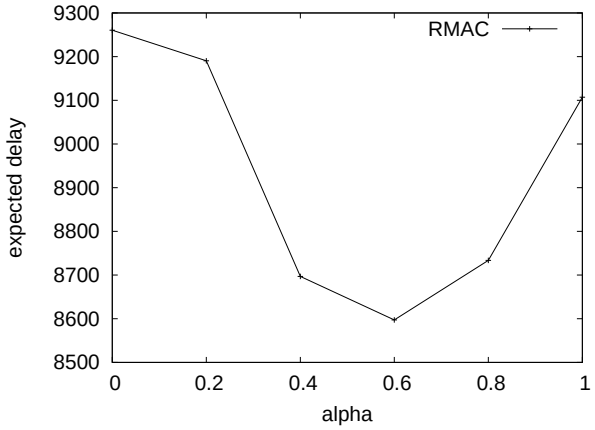

Fig. 2. Variation of delay with α for bursty traffic

a) Low arrival rates: Figure 3 compares the performance of these algorithms for random traffic at low arrival rates. As expected, our algorithm performs much better than the one in [3], since the latter has an aggressive strategy for increasing the number of minislots, which increases delay. Both DQRUMA and RMAC are fairly close to the ideal algorithm, because they use fewer minislots. We remind the reader that the Ideal algorithm is centralized and hence can resolve contention with no delay.

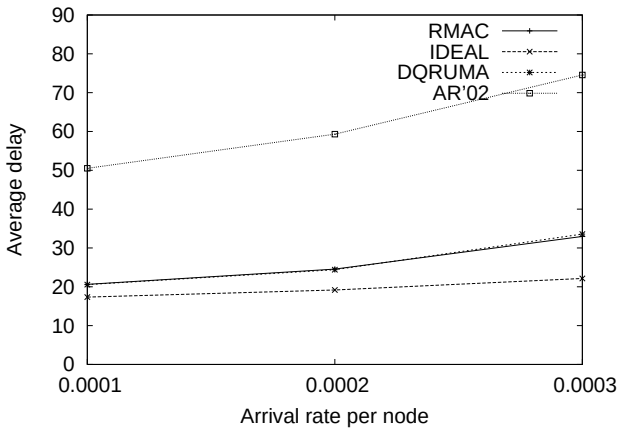

Fig. 3. Performance of our protocol for low arrival rates (random traffic)

b) High arrival rates: Figure 4 shows the performance of all four algorithm for high arrival rates. In this case, RMAC and the algorithm in [3] do significantly better than DQRUMA, since they are far more aggressive in increasing the number of minislots. We note that for heavy traffic, most nodes are always transmitting packets and thus utilize piggybacked requests to avoid the contention phase. This implies that the performance of RMAC and the algorithm in [3] are close to that of the Ideal algorithm in this scenario. Thus, RMAC combines the advantages of DQRUMA at low arrival rates and of the algorithm in [3] at high arrival rates.

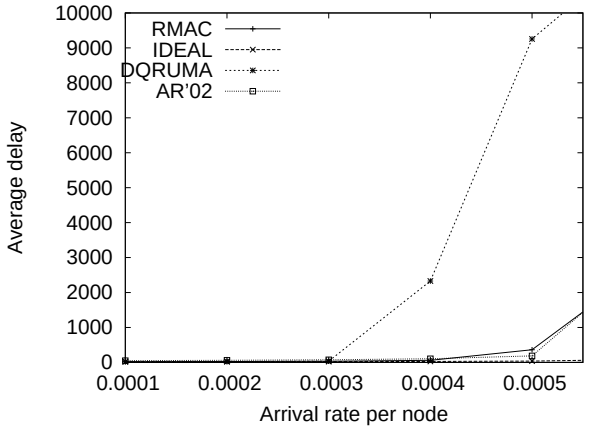

Fig. 4. Performance of our protocol for all arrival rates (random traffic)

c) Bursty arrivals: In Figure 5, we compare the performance of the same algorithms for random bursty arrivals. Due to the bursty nature of the traffic, we expect that the sensor nodes will be non-empty more often and so the piggybacked reservations avoid the overhead of the contention subsystem, as noted in [2]. Thus all the algorithms work well in general in this case. In addition, RMAC and the algorithm in [3] do significantly better than DQRUMA at very high arrival rates. Since all algorithms with piggybacked reservations do better with bursty traffic, all the algorithms perform well with ON-OFF bursty traffic. We omit the graphs for those results.

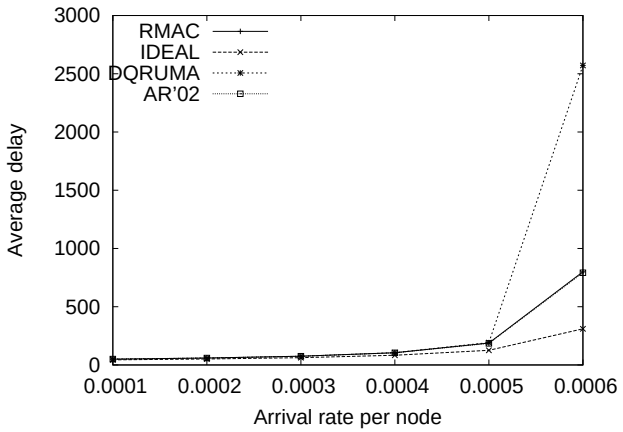

Fig. 5. Performance of our protocol for bursty traffic

C. Performance-fairness tradeoff

We note that this paper does not attempt to solve the problems due to the capture effect. In general, the capture effect increases performance but also increases unfairness. Therefore, in our simulations, we did not model the capture effect.

In Figure 6, we show how the performance of RMAC degrades with the parameter f. The degradation of performance of DQRUMA and the algorithm in [3] as the parameter f is varied are similar and are not shown.

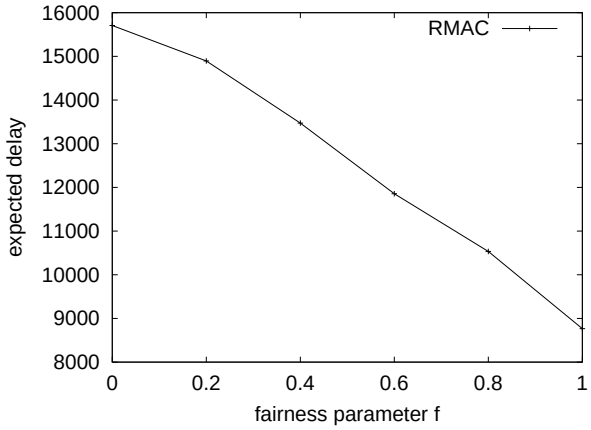

Fig. 6. Degradation of the performance of RMAC with fairness parameter f

VI. Conclusions and Future Work

In this paper we presented a simple, randomized algorithm for medium access control in heterogeneous sensor networks. We presented simulation results to show that our algorithm performs better than several existing demand assignment MAC algorithms. In addition, we examine several problems involved in the design of efficient demand assignment algorithms, and propose solutions for each of those problems. While we make no claims about the optimality of our solutions, we believe that our work highlights and motivates the problems concerned.

We are currently investigating the use of better prediction algorithms for this problem. In addition, we are working on a generalization of RMAC for general wireless networks that support QoS requirements.

Acknowledgments

We thank the anonymous referees of this paper for several excellent suggestions.

References

[1] D. Jeong, C. Hoi, and W. Jeon, "Design and performance evaluation of a new medium access control protocol for local wireless data communications," *IEEE/ACM Transactions on Networking*, vol. 3, no. 6, pp. 742–752, 1995.

[2] M. Karol, Z. Liu, and K. Eng, "Distributed-queueing request update multiple access (DQRUMA) for wireless packet (ATM) networks," in *Proceedings of ICC'95*, 1995, pp. 1224–1231, seattle, USA.

[3] M. Abolelaze and R. Radhakrishnan, "The performance of a new wireless MAC protocol," in *3G Wireless 2002*, 2002, available at http://www.cs.yorku.ca/~aboelaze/publication/AbR02.pdf.

[4] "Crossbow technology inc." http://www.xbow.com/.

[5] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of INFOCOM 2002*, June 2002.

[6] V. Rajendran, K. Obraczka, and J. Garcia-Luna-Aceves, "Energy-efficient, collision-free medium access control for wireless sensor networks," in *Proceedings of ACM SenSys 03*, November 2003.

[7] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, pp. 102–114, August 2002.

[8] C.-Y. Chong and S. Kumar, "Sensor networks: Evolution, opportunities, and challenges," *Proceedings of the IEEE*, August 2003.
[9] D. Estrin, D. Culler, and K. Pfister, "Connecting the physical world with pervasive networks," *IEEE Journal of Pervasive Computing*, vol. 1, no. 1, January-March 2002.
[10] N. Xu, "A survey of sensor network applications," 2004, available at http://enl.usc.edu/~ningxu/papers/survey.pdf.
[11] A. Chandra, V. Gummalla, and J. O. Limb, "Wireless medium access control protocols," *IEEE Communications Surveys*, vol. Second Quarter, pp. 2–15, 2000.
[12] P. Baldi, R. Jurdak, and C. V. Lopes, "A taxonomy of medium access control protocols for ad hoc networks," *IEEE Communications Surveys*, vol. First Quarter, pp. 2–16, 2004.
[13] C. E. Jones, K. M. Sivalingam, P. Agrawal, and J. C. Chen, "A survey of energy efficient network protocols for wireless networks," *Wireless Networks*, vol. 7, no. 4, pp. 343–358, 2001.
[14] Y.-K. Kwok and V. Lau, "Performance evaluation of multiple access control schemes for wireless multimedia services," *IEE Proceedings-Communications*, vol. 148, no. 2, pp. 86–94, April 2001.
[15] T. van Dam and K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *Proceedings of the first international conference on Embedded networked sensor systems*. ACM Press, 2003, pp. 171–180.
[16] G. Lu, B. Krishnamachari, and C. Raghavendra, "An adaptive energy-efficient and low-latency MAC for data gathering in sensor networks," in *4th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN 04)*, April 2004.
[17] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He, "RAP: A real-time communication architecture for large-scale wireless sensor networks," in *IEEE RTAS*, 2002.
[18] L. C. Zhong, R. Shah, C. Guo, and J. Rabaey, "An ultra-low power and distributed access protocol for broadband wireless sensor networks," in *IEEE Broadband Wireless Summit*, May 2001.
[19] J. Mikkonen, J. Aldis, G. Awater, A. Lunn, and D. Hutchinson, "The magic WAND - functional overview," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 6, pp. 953–972, August 1998.
[20] K. Jamieson, H. Balakrishnan, and Y. C. Tay, "Sift: A MAC protocol for event-driven wireless sensor networks," MIT, Tech. Rep., may 2003. [Online]. Available: http://www.lcs.mit.edu/publications/pubs/pdf/MIT-LCS-TR-894.pdf
[21] T. Ozugur, M. Naghshineh, P. Kermani, C. M. Olsen, B. Rezvani, and J. A. Copeland, "Balanced media access methods for wireless networks," in *Mobile Computing and Networking*, 1998, pp. 21–32.
[22] T. Nandagopal, T. eun Kim, X. Gao, and V. Bharghavan, "Achieving mac layer fairness in wireless packet networks," in *MOBICOM*, 2000, pp. 87–98.
[23] C. E. Koksal, H. Kassab, and H. Balakrishnan, "An analysis of short-term fairness in wireless media access protocols (poster session)," in *Proceedings of the 2000 ACM SIGMETRICS*. ACM Press, 2000, pp. 118–119.
[24] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed. Wiley InterScience, 2001.
[25] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 147–163, 2002.
[26] R. G. Gallagher, *Discrete Stochastic Processes*. Kluwer Academic Publishers, 1996.
[27] K. K. Ramakrishnan and H. Yang, "The ethernet capture effect: Analysis and solution," in *Proceedings of the IEEE 19th Local Computer Networks Conference*, October 1994.
[28] E. Tobagi and L. Kleinrock, "Packet switching in radio channels: Part II - the hidden terminal problem in carrier sense multiple.access modes and the busy-tone solution," *IEEE Transactions on Communications*, vol. COM- 23, no. 12, pp. 1417–1433, 1975.
[29] L. Kleinrock, *Queueing systems*. Wiley, New York, 1976.

APPENDIX

COMBINATORIAL RESULTS USED IN PROOFS

Lemma 1: The number of ways of throwing x balls in y bins, $x \geq 2y$ such that every bin has at least 2 balls is given by

$$T(x,y) = \binom{x-y-1}{y-1}.$$

Proof: First reserve $2y$ balls which will be distributed equally among y bins. Place the remaining $x-2y$ balls into y bins. This can be done in $\binom{x-y-1}{y-1}$ ways. □

Lemma 2: Suppose m balls are thrown into n bins. Let n_e be the number of bins with no balls, n_s be the number of bins with exactly one ball and n_c be the number of bins with more than one ball. Then the probability that $n_s = j, n_e = k$ is given by

$$\Pr(n_s = j, n_c = k, n_e = n-j-k | m, n) = \frac{1}{n^m}\binom{n}{j,k}\binom{m-j-k-1}{k-1}$$

Proof: The j bins with exactly one and the k bins with more than one balls are chosen in $\binom{n}{j,k}$ ways. j of the balls can be placed in the j bins, one in each, in exactly 1 way. The remaining $m-j$ balls can be placed in k bins such that every bin has at least two balls in $T(m-j,k)$ ways where $T()$ was defined in Lemma 1. To get the probability, we simply divide by the number of possible ways of putting m balls in n bins.

$$\begin{aligned}\Pr(n_s &= j, n_c = k, n_e = n-j-k | m, n) \\ &= \frac{1}{n^m}\binom{n}{j,k}T(m-j,k) \\ &= \frac{1}{n^m}\binom{n}{j,k}\binom{m-j-k-1}{k-1} \text{ by Lemma 1}\end{aligned}$$

□

Lemma 3: Suppose m balls are thrown into $n = Cm$ bins. Let n_s be the number of bins with exactly one ball. Then its expected value, $E[n_s]$, is given by $E[n_s] \approx 1/e^{1/C}$.

Proof: The probability that any bin has exactly 1 ball is given by $p = \frac{1}{n}(1-1/n)^{m-1}$. Therefore the expected number of bins that have exactly 1 ball is $E[n_s] = np = (1-1/n)^{m-1} = (1-1/Cm)^{m-1} \approx 1/e^{1/C}$. □

A SIMPLE EXPRESSION FOR THE MAXIMUM LIKELIHOOD ESTIMATOR

Lemma 4: The Maximum Likelihood estimator of the number of nodes $m(t)$ is given by

$$\widehat{m}(t) = \text{MLE}[m(t)] = n_s(t) + 2n_c(t).$$

Proof: Define $L(x) = \Pr(n_s = j, n_c = k, n_e = n-j-k | m = 2n_c + n_s + x, n)$ and $L(x+1) = \Pr(n_s = j, n_c = k, n_e = n-j-k | m = 2n_c + n_s + x + 1, n)$ where $x = 0, 1, 2, \ldots$.

From Lemma 2,

$$\begin{aligned} L(x) &= \frac{1}{n^{2n_c+n_s+x}}\binom{n}{n_s, n_c}\binom{x+n_c-1}{n_c-1} \\ L(x+1) &= \frac{1}{n^{2n_c+n_s+x+1}}\binom{n}{n_s, n_c}\binom{x+n_c}{n_c-1} \\ \frac{L(x)}{L(x+1)} &= n\binom{x+n_c-1}{n_c-1}/\binom{x+n_c}{n_c-1} \\ &= \frac{n(x+1)}{x+n_c} \end{aligned}$$

Since $n \geq 1$, $nx \geq x$ and by definition $n \geq n_c$, so $nx + n = n(x+1) \geq x + n_c$. Therefore, $\frac{L(x)}{L(x+1)} \geq 1$, and so $L(x) \geq L(x+1)$. Since this is true for all x, it follows that $L(0) \geq L(1) \geq L(2) \ldots$, so that the Maximum Likelihood estimator corresponds to $x = 0$. The lemma follows. □