

# DRAND: Distributed Randomized TDMA Scheduling For Wireless Ad-hoc Networks

Injong Rhee, Ajit Warrier, Jeongki Min  
Dept of Computer Science  
North Carolina State University  
Raleigh, NC 27695  
rhee,acwarrie,jkmin@ncsu.edu

Lisong Xu  
Dept of Comp. Sci. and Eng.  
University of Nebraska  
Lincoln, Nebraska 68588  
xu@cse.unl.edu

## ABSTRACT

This paper presents a distributed implementation of RAND, a randomized time slot scheduling algorithm, called DRAND. DRAND runs in  $O(\delta)$  time and message complexity where  $\delta$  is the maximum size of a two-hop neighborhood in a wireless network while message complexity remains  $O(\delta)$ , assuming that message delays can be bounded by an unknown constant. DRAND is the first fully distributed version of RAND. The algorithm is suitable for a wireless network where most nodes do not move, such as wireless mesh networks and wireless sensor networks. We implement the algorithm in TinyOS and demonstrate its performance in a real testbed of Mica2 nodes. The algorithm does not require any time synchronization and is shown to be effective in adapting to local topology changes without incurring global overhead in the scheduling. Because of these features, it can also be used even for other scheduling problems such as frequency or code scheduling (for FDMA or CDMA) or local identifier assignment for wireless networks where time synchronization is not enforced.

## Categories and Subject Descriptors

C.2.1 [Network Architecture and Design]: Wireless communication

## General Terms

Experimentation, Performance, Algorithms

## Keywords

Wireless Sensor Networks, Medium Access Control, Network Performance

## 1. INTRODUCTION

Is TDMA better than CSMA for wireless networks or vice versa? We don't know the answer to that question as it is an

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MobiHoc'06, May 22–25, 2006, Florence, Italy.

Copyright 2006 ACM 1-59593-368-9/06/0005 ...\$5.00.

ill-posed question. We will tell you why later though. This paper is not about whether TDMA is better than CSMA or vice versa. Nonetheless, we are biased to claim that TDMA has a unique place in the MAC protocol design space for wireless networks, particularly where most of nodes can be assumed to be stationary such as wireless mesh networks and sensor networks. The pros and cons of TDMA have been discussed in earlier papers (e.g. [24]). Let us sum up some of these arguments. TDMA uses topology information as a basis for access scheduling and this topology information comes in the form of neighboring and interference relations among nodes. Based on tight synchrony among their neighbors, time slots are scheduled for access in such a way that no two interfering nodes access the channel at the same time. However, TDMA strongly ties its performance to the accuracy of this information and synchrony. When the accuracy can be upheld by the system, TDMA can deliver very good performance especially under high contention. Unfortunately, wireless networks do not make it easy to maintain this accuracy. First, it is very difficult to precisely capture this interference relation because of interference range irregularity [25]. Second, channel conditions may change interference relations among nodes over time. Third, clocks can drift and tight synchronization may incur too much overhead.

On the other hand, CSMA does not use any topology or clock information. Thus it is highly robust to any change in the network. But in wireless networks, some of this information, although in a less accurate or reliable manner, is available to the nodes at little cost. But since CSMA does not make use of this information, contention among neighboring nodes must be resolved for every data transmission. Thus its performance under high contention suffers because of high overhead in resolving contention and collision [13].

We show, through a protocol called Z-MAC [21], that the information about time and interference relation in wireless networks, although imprecise, can be effectively used to improve the performance of CSMA under high contention. Z-MAC, behaves like CSMA under low contention, and like TDMA under high contention. When the interference relation and synchrony become less accurate, its performance gracefully degrades to that of pure CSMA; thus its performance is always in between that of CSMA and TDMA depending on the accuracy of the information. It is also shown that even under no clock synchronization, use of imprecise information on local topology<sup>1</sup> results in higher throughput

<sup>1</sup>each node needs to know only about its two-hop neighbors.

than CSMA under high contention. This is because of the topology and time information, although imprecise, available by the TDMA schedule.

Z-MAC relies on an efficient static time slot scheduling algorithm. In this paper we present a new distributed randomized time slot assignment algorithm, called *DRAND*. Optimal static time slot scheduling is NP-hard [20]. A heuristic, but centralized solution, called RAND, is known to give very efficient slot schedules [20]. As far as we know, DRAND is the first distributed version of RAND. There have been heuristic distributed approximation to RAND [4, 27, 22] wherein RAND has also been used as a performance benchmark. Assuming that the message delay is within some unknown constant bounds, DRAND incurs  $O(\delta)$  running time and message complexity where  $\delta$  is the number of two-hop neighbors. As the bounds are unknown, we don't use actual values of the message delays in the algorithm, but we use it only for analysis. Since the algorithm may need to run periodically to handle node mobility and new node joining, the low complexity of the algorithm is always desirable. The performance of DRAND is scalable to partial change in network topology because DRAND can re-compute the schedule without involving global changes. DRAND does not require any time synchronization to run although TDMA using the schedule requires it. This feature is useful because the algorithm can be used for other scheduling such as frequency or code scheduling (FDMA or CDMA) or local identifier assignment [10] where clock synchronization is not provided. We have implemented DRAND in TinyOS and in NS and measured the performance in a real testbed of 42 Mica2 nodes deployed over faculty and student offices of our department building. We also implement a TDMA protocol in TinyOS to demonstrate that the schedule obtained by DRAND can be far more efficient in resolving contention than those produced by existing scheduling algorithms.

**Disclaimer.** So, is TDMA better than CSMA? As far as we are concerned, the question is ill-posed as TDMA and CSMA can co-exist complementing each other under certain circumstances. But the question has been a subject of debate for a long time and will likely remain so because both paradigms separately are popular in commercial standards and applications (e.g., for TDMA [3, 2], for CSMA [7]). This paper is not about this question, but rather, assuming that TDMA is useful, it improves the state of art in the TDMA domain. Thus, our performance study is confined only in the context of TDMA and no comparison with CSMA or its kind is provided.

## 2. TDMA SLOT ASSIGNMENT

In this section, we formally define the slot assignment problem. The network is represented by a graph  $G = (V, E)$  where  $V$  is the set of nodes, and  $E$  is the set of edges. An edge  $e = (u, v)$  exists if and only if  $u$  and  $v$  are in  $V$  and  $u$  and  $v$  can hear each other (i.e., all edges are bidirectional). We relax this requirement in Section 5. Time is slotted into a non-overlapping equal time period called *time frame* which is also divided into *MaxSlot* non-overlapping equal time periods call *time slots*. Time slots are numbered from 1 to *MaxSlot*, assuming that *MaxSlot* is sufficiently large enough to handle all the assignment strategies for any input graph.

We say that two nodes  $u$  and  $v$  are in *conflict* if and only if the simultaneous transmission from  $u$  and  $v$  causes radio in-

terference at some node. In the broadcast scheduling mode, conflict can happen among all the nodes within a two-hop distance. In the unicast mode, conflict happens among all the nodes within a one-hop distance of a transmitter and a receiver. A good list of conflict relations in wireless networks is defined in [20].

We formally define the *slot assignment problem* as finding a time slot for each node, given an input graph  $G$  and conflict definition, such that if any two nodes are in conflict, they do not have the same time slot. The performance of an algorithm for the slot assignment problem can be determined by three quantities:

**Maximum Slot Number:** Note that the TDMA slot assignment problem is a direct extension of the graph coloring problem, where the goal is to color the vertices of a graph with minimum number of colors such that no two adjacent nodes have the same color. (One can see this by connecting each pair of conflicting nodes by an edge – then coloring this new graph is analogous to finding a TDMA schedule, since no two conflicting nodes can share the same TDMA slot). The graph coloring (and hence TDMA assignment) problem is known to be NP-Hard [20]. Hence, heuristic solutions often report the maximum number of colors required to obtain a coloring assignment for all executions of the given coloring algorithm. We will henceforth call this the *maximum slot number*, also called the *worst-case chromatic number* in the graph coloring terminology.

**Running time:** the maximum time taken for all the nodes in  $V$  to decide on their time slots for all executions of the algorithm.

**Message complexity:** the maximum number of messages transmitted for all the nodes in  $V$  to decide on the time slots for all executions of the algorithm.

## 3. DISTRIBUTED RAND

In this section we describe the DRAND TDMA slot assignment algorithm. In the description, we assume the broadcast mode. Our description can be easily extended to other conflicting relations. DRAND is a distributed implementation of RAND [20], a centralized slot assignment algorithm, and thus achieves the same channel efficiency as RAND, but with  $O(\delta)$  average time and message complexity where  $\delta$  is the maximum number of two-hop neighbors for any node in the network.

### 3.1 Algorithm Specification

DRAND runs in rounds. The duration of each round is adjusted dynamically depending on the estimates of network delays. However, the algorithm does *not* require each node to synchronize on round boundary. There are four states that a node maintains: IDLE, REQUEST, GRANT, and RELEASE. Figure 1 shows the state diagram for our DRAND implementation.

Initially a node  $\mathbf{A}$  is in the IDLE state. During the IDLE state,  $\mathbf{A}$  tosses a coin whose probability of getting *head* or *tail* is  $1/2$ . If a node gets *head*, it runs a lottery that has some preset probability of success. If it wins the lottery, it negotiates with its neighbor to select a time slot by exchanging messages. More precisely, it runs as follows. Each

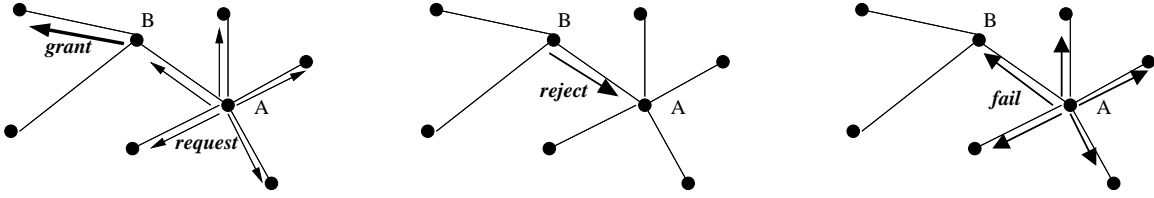


Figure 2: A failed round because a node **B** has sent a *grant* message to another one-hop neighbor before receiving the *request* from **A**.

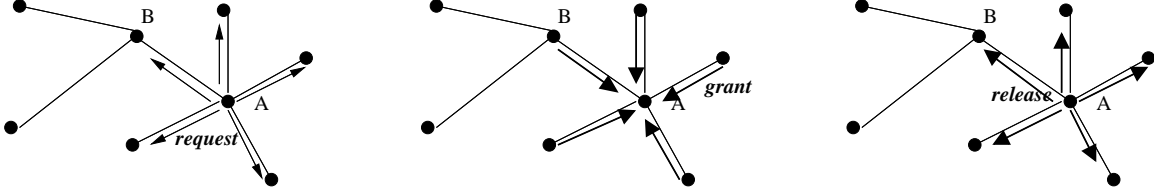


Figure 3: A successful round where **A** decides on a time slot after receiving *grant* messages from its one-hop neighbors.

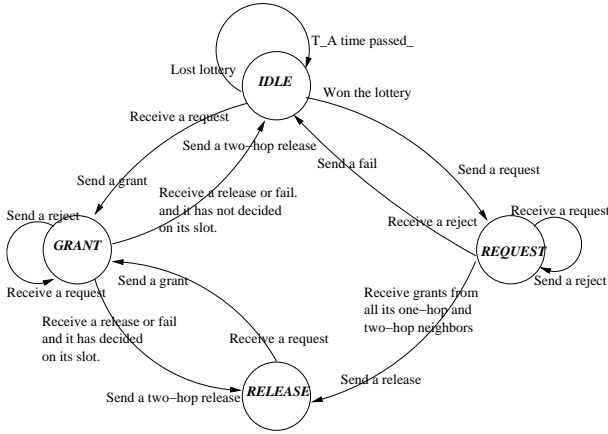


Figure 1: The state diagram of DRAND. The statement at the beginning of an arrow is the condition that makes the indicated state transition and that at the end of the arrow is the action taken before moving to a new state.

node  $j$  maintains estimates  $C_j$  on its one-hop and two-hop neighbors who have not decided on their own slots. If it has been more than a  $T_A$  time since it tried the lottery last time, **A** runs the lottery for which its winning chance is set to a probability  $p_A = 1/k$  where  $k$  is set to the maximum of  $C_j$  for all  $j$  in the one-hop and two-hop neighbors of **A**.

Let  $i$  be the number of times that **A** has tried the lottery so far and we say that **A** is in the  $i$ -th round. If it wins the lottery, **A** moves to the REQUEST state, and broadcasts a  $request_i$  message to its one-hop neighbors. If it loses the lottery, it remains in the IDLE state.  $T_A$  is set to  $3d_A$  where  $d_A$  is **A**'s approximate estimate on the maximum one-way message delay (this estimate is obtained whenever **A** receives a response from other nodes to its earlier requests; explained below).

When a neighbor **B** receives a  $request_i$  from **A**, if **B** is in the IDLE or RELEASE state, it changes into the GRANT

state and sends a  $grant_i$  message to **A**. **B** is in these states only when (1) no neighbors of **B** have sent a *request* (note that it is also possible that **B** might be in some round different from **A**'s because **A** and **B** are not synchronized); (2) **B** has not sent any *grant* to any node so far; or (3) if from a node other than **A** that **B** has sent a *grant* to, it has received a *fail* or *release* (defined below). When sending a  $grant_i$ , **B** includes in that message the time slots that are chosen by its one-hop neighbors (it knows this information when it receives a *release* message from its neighbors).

When receiving a  $request_i$  from **A**, if **B** is in the REQUEST or GRANT state, then **B** sends a  $reject_i$  message to **A**. When receiving a  $reject_i$  from any node, **A** sends a  $fail_i$  message to all its one-hop neighbors and changes its state to the IDLE state. When **B** receives a  $fail_i$  from **A** whose  $request$  turned **B**'s current state to GRANT, **B** returns to the IDLE state if it has not decided on its slot already, or to the RELEASE state if it has decided on its slot. Figure 2 illustrates a failed round because a node **B** has sent a *grant* to its another one-hop neighbor before receiving the *request* from **A**.

If **A** does not receive any  $grant_i$  or  $reject_i$  from a one-hop node **B** within some time  $d_A$ , **A** retransmits  $request_i$  to those nodes that it has not received a  $grant_i$  from. When a node receives a  $request_i$  from another node for which it has already sent a  $reject_i$ , then it retransmits the  $reject_i$  to that node. Whenever **A** receives a  $grant_i$  or  $reject_i$ , it estimates the message delay by taking differences of time stamps of  $request_i$  and the received message. If the new estimate is larger than the current value of  $d_A$ ,  $d_A$  is set to the new estimate.

As **A** receives a  $grant_i$  from its entire one-hop neighbors for  $request_i$ , it decides on its time slot to be the minimum of the time slots that have not been taken by its two-hop neighbors before this round (this information is known through the slot information piggy-backed in the *grant* messages). Then **A** enters the RELEASE state and broadcasts a  $release_i$  message containing information about its selected time slot to its one-hop neighbors. Figure 3 illustrates a successful round.

On receiving a  $release_i$  message, a one-hop neighbor of **A** turns into the IDLE state if it has not decided on its slot or the RELEASE state otherwise, and re-broadcasts that  $release_i$  message to its one-hop neighbors. Let us call this forwarded  $release_i$  message a *two-hop-release<sub>i</sub>* message. When nodes receiving *two-hop-release<sub>i</sub>* messages, they can estimate  $k$ . If a node **B** does not receive a  $fail_i$  or  $release_i$  message after sending  $grant_i$  to **A** for time  $d_B$ , it retransmits  $grant_i$ . If a node receives a  $grant_i$  for which it has previously sent a  $fail_i$  or  $release_i$ , then it retransmits the  $fail_i$  or  $release_i$  to that node.

## 4. DRAND ANALYSIS

In this section we analyze the performance and correctness of the DRAND algorithm. We show that the schedule created by DRAND is a valid conflict-free TDMA schedule. We then show that DRAND gives the same maximum slot number as RAND, a centralized TDMA slot assignment algorithm. Finally, we analyze the time and message complexity of DRAND.

### 4.1 DRAND Correctness

**THEOREM 4.1.** *The execution of DRAND results in a conflict free TDMA schedule.*

**Proof:** To prove the validity of the TDMA schedule created by the execution of DRAND, it is enough to prove that no two nodes within two hops of each other select the same time slot. This can be easily seen from the followings: (1) a node has to receive *grant* messages from all of its one-hop neighbors, (2) any nodes that are two-hop away from each other, share at least one common one-hop neighbor and (3) a node sends at most one *grant* message at each round. The *grant* messages contain the list of slots already taken by their one-hop neighbors. This property ensures that when a node decides, it always picks the minimum time slot that is not taken by two-hop neighbors and no other nodes within two-hop neighbors can pick the same time slot. ■

### 4.2 Equivalence with RAND

In RAND, nodes are sorted in a random total order, and each node is assigned, in that order, the minimum time slot which has not been taken by its conflicting, but preceding nodes. This requires knowledge of global network topology. DRAND, on the other hand, achieves the same maximum slot number as RAND, but requires only local (two hop) topology information.

**THEOREM 4.2.** *For any execution  $E_r$  of RAND, there exists a corresponding execution of DRAND that produces the same slot assignment as  $E_r$ , and conversely, for any execution of  $E_d$  of DRAND, there exists a corresponding execution of RAND that gives the same slot assignment as  $E_d$ .*

**Proof:** Suppose that  $S (= v_1, v_2, v_3, \dots, v_n)$  is the sequence in which the nodes are assigned the time slots in  $E_r$ , and  $s(u)$  is the slot assigned to a node  $u$  in  $E_r$  by RAND (Here,  $n = |V|$ ). We divide  $S$  into  $m$  non-overlapping partitions  $P_1, P_2, P_3, \dots, P_i, P_{i+1}, \dots, P_m$  for  $m > 0$  as follows. (1)  $P_i$  is a subsequence of  $S$ , (2) the concatenation of  $P_1$  through  $P_m$  yields  $S$  and (3) no two nodes in each partition  $P_i (1 \leq i \leq m)$ , conflict with each other. That is,  $S = P_1 P_2 P_3 \dots P_m = v_1, v_2, v_3, \dots, v_n$ . The following lemmas are sufficient to prove the first part of the theorem.

**LEMMA 4.1.** *There exists an execution  $E_d$  of DRAND that all the nodes in  $P_i (1 \leq i \leq m)$  decide their time slots in a round  $i$ .*

**Proof:** We will prove the lemma by induction. Consider  $i = 1$ . Each node in  $P_1$  sends *requests* and any conflicting nodes of nodes in  $P_1$  do not send a *request* at the first round. Thus all neighboring nodes of  $P_1$  are not conflicting with  $P_1$  and they send *grant* messages to  $P_1$  in their first round. Such execution is possible by DRAND. This guarantees that nodes in  $P_1$  decide their own slots in the first round because no two nodes in  $P_1$  are in conflict by the definition of partition. By the induction hypothesis, suppose that there exists an execution of DRAND in which nodes in  $P_i$  decide their time slots in a round  $i (1 \leq i \leq h - 1)$ . That is,  $P_1 P_2 \dots P_{h-1}$  is the node ordering up to round  $h - 1$  in some execution of DRAND. As the nodes in  $P_h$  have not decided in the earlier rounds than  $h$ , according to the algorithm, there exists a non-zero probability that only those nodes in  $P_h$  send a *request* in round  $h$ . Since by definition, those nodes are not in conflict, they decide on their slots in round  $h$ . ■

**LEMMA 4.2.** *In the execution  $E_d$  of DRAND, a node  $v_i (v_i, 1 \leq i \leq n)$  chooses the same slot as in  $E_r$  (i.e.,  $s(v_i)$ ).*

**Proof:** By induction on  $i$ . Consider  $i = 1$ . Since  $v_1$  is the first node to decide in  $E_r$  and  $E_d$ , it will choose time slot 1 in both executions. Suppose that all the nodes before  $v_h$ ,  $(v_1, \dots, v_{h-1})$  choose the same colors in both executions. By the algorithm of RAND, if  $v_h$  chooses  $s(v_h)$ , then  $s(v_h)$  must be the minimum slot that has not taken by all of its conflicting nodes that have decided earlier than  $v_h$ . Since in both executions,  $v_h$  will have the same set of conflicting nodes that have decided before  $v_h$  by Lemma 4.1, and by the hypothesis, they have chosen the same colors in both executions. Therefore, the algorithm of DRAND will also dictate  $v_h$  to choose  $s(v_h)$ . ■

Lemmas 4.1 and 4.2 prove the first part of the Theorem 4.2. The proof for the second part of Theorem 4.2 is similar and we omit the proof. ■

### 4.3 Complexity Analysis

Each node  $i$  in the network runs DRAND in rounds of time period  $T_i$ .  $T_i$  is set to  $3d_i$ , where  $d_i$  is an estimate of one-way message delay by node  $i$ . Nodes are *not* required to be synchronized on round boundaries.

In order for a node  $i$  to finish slot selection,  $T_i$  should be sufficiently long to (a) send a request message, (b) receive grant messages from all its one hop neighbors, (c) select the minimum time slot available and (d) send a release message. Initially, each node  $i$  starts with a default value of  $T_i$ .

We define  $d_{max}$  and  $d_{min}$  to be the maximum and minimum one-way message delays in the network. Let  $\Delta = \lceil T_{max}/T_{min} \rceil$  and  $T_{max}$  be  $3d_{max}$ . Ignoring the time taken to execute the internal steps of running the lottery and choosing the slot number, every node must try the lottery at least once and at most  $\Delta$  times in  $T_{max}$  time. We define the time period of  $T_{max}$  to be a *super-round* or in short *s-round*.

**THEOREM 4.3.** *Assuming the maximum and minimum message delays are bounded by some constants, the expected number of s-rounds for a node to acquire a time is less than*

$2(\delta + 1) \cdot e^{0.5\Delta}$ , and the probability that it takes longer than some constant factor  $c > 1$  times the expected, is less than or equal to  $1/e^c$ .

**Proof:** Let the *contenders* of  $j$ , denoted  $C(j, k)$ , be the set of nodes conflicting with  $j$  (within two hops of  $j$ ), and who have not yet acquired their time slots by  $k$ -th s-round. During the  $k$ -th s-round, a contender  $i \in C(j, k)$  may try the lottery at least once. Let  $L(j, k)$  denote the event that node  $j$  wins the lottery in the  $k$ -th s-round.

Node  $j$  acquires a slot if it wins the lottery in the  $k$ -th s-round and no other contender wins the lottery during that s-round. Since contenders  $i \in C(j, k)$  can have at most  $\Delta$  lottery tries during an s-round period  $T_{max}$ , the probability  $Pr(j \text{ leave}, t)$  that node  $j$  acquires a slot in the  $t$ -th s-round is bounded as follows:

$$Pr(j \text{ leave}, t) \geq Pr(L(j \text{ leave}, 1)) \quad (1)$$

$$\geq Pr(L(j, 1)) \prod_{i \in C(j)} \prod_{k=1}^{\Delta} (1 - Pr(L(i, k))) \quad (2)$$

$$\geq Pr(L(j, 1)) \prod_{i \in C(j)} (1 - Pr(L(i, \Delta)))^{\Delta} \quad (3)$$

$$\geq \frac{1}{2(\delta + 1)} \prod_{i \in C(j)} \left(1 - \frac{1}{2(|C(j)| + 1)}\right)^{\Delta} \quad (4)$$

$$= \frac{1}{2(\delta + 1)} \left(1 - \frac{1}{2(|C(j)| + 1)}\right)^{|C(j)|\Delta} \quad (5)$$

$$> \frac{1}{2(\delta + 1)} \left(\frac{1}{\sqrt{e}}\right)^{\Delta} \quad (6)$$

Eq. 3 is because  $Pr(L(i, \Delta)) \geq Pr(L(i, k))$  for  $1 \leq k \leq \Delta$ . Eq. 4 is because of the followings: (1)  $Pr(L(i, \Delta)) \leq 1/(2(|C(j)| + 1))$  because contender  $i$  uses, in setting  $p_i$ , the inverse of the maximum of the neighbor sizes of its contender set which includes  $j$  (so at the minimum,  $|C(j)| + 1$ ), and  $C(j)$  does not change while  $j$  selects the slot and  $1/2$  because of the coin-tossing before the lottery, (2)  $Pr(L(j, 1)) \geq 1/(2(\delta + 1))$  because  $\delta$  is the maximum contender set size of any node in the network. Eq. 6 is because  $(1 - \frac{1}{2(|C(j)| + 1)})^{|C(j)|} > 1/\sqrt{e}$ .

Since the above result is independent of  $j$  and  $k$ , it gives the lower bound on the probability that a node acquires a slot in any s-round. Let  $M$  be the random number representing the number of s-rounds before a node acquires a slot with the lower bound probability.  $M$  clearly has a geometric distribution.

$$Pr(M = k) = P_{low}(1 - P_{low})^{k-1} \quad (7)$$

where  $P_{low} = \frac{1}{2(\delta + 1)e^{0.5\Delta}}$ .

From the above, we can obtain the upper bound on the expected number of s-rounds that a node takes to acquire a slot as follows:

$$E[M] = \frac{1}{P_{low}} = 2(\delta + 1) \cdot e^{0.5\Delta} \quad (8)$$

Finally, we calculate the probability that a node does not acquire a slot for a period  $c$  times longer than the expected:

$$\begin{aligned} Pr(M > c \cdot E[M]) &= \sum_{k=c \cdot E[M]+1}^{\infty} P_{low}(1 - P_{low})^{k-1} \quad (9) \\ &= (1 - P_{low})^{c \cdot E[M]} \quad (10) \\ &= \left(1 - \frac{1}{E[M]}\right)^{c \cdot E[M]} \leq \frac{1}{e^c}. \quad (11) \end{aligned}$$

■

The above analysis assumes that message delays are bounded by a constant. In reality, depending on the MAC protocol being used to implement DRAND, message delays of a node could be a function of the size of its neighborhood since neighboring nodes compete to access the common channel. In CSMA, it is especially so while in CDMA, it may not be the case. However, we don't have a clear bound on this message delay because this highly depends on the MAC protocol being used and there is no asymptotic analysis on the complexity of this message delay for a given MAC protocol. For such networks where message delays cannot be bounded by a constant, we do not claim that our analysis is directly applicable. In this paper, we rely on experimental work to estimate the performance of DRAND for such networks. As we shall see next section, the average number of rounds even in such a network is closely approximated to our analysis.

**THEOREM 4.4.** *Assuming that the message delay is bounded by a constant, DRAND has an expected message complexity of  $O(\delta)$ .*

**Proof:** In one round, a contender can try the lottery for  $\Delta$  times. In each try, it can be a winner and gets rejected, thus sending  $O(1)$  messages. Therefore, in one round, it can send  $O(\Delta)$  messages. Since there are  $O(\delta)$  rounds on average, each process can send  $O(\Delta \cdot \delta)$  messages on average. ■

## 5. DISCUSSION

The practical implementation of DRAND in a real wireless network must deal with several practical, yet important technical issues. These issues arise mainly from packet losses, communication asymmetry, and node and communication failures that our system model does not capture. These anomalies can severely hamper the progress of DRAND possibly causing deadlocks if they are not handled properly. This section discusses our approaches to these issues. In Section 6, we also demonstrate their efficacy by a real implementation and performance evaluation of DRAND in a wireless sensor network testbed.

Our model in Section 4 assumes that all messages are delivered in a bounded time. However, we claim that the correctness of the algorithm is still ensured even under weaker timing models where messages are delivered within a finite time (instead of a bounded time). Note that if a message cannot be delivered between two nodes even after an infinite number of retransmissions, then the two nodes are not connected (i.e., they are not in a one hop distance). However, in these weaker models, the running time of the algorithm cannot be bounded in real-time.

The description of the algorithm handles packet losses via retransmission. However it is possible that some communi-

cation links become unavailable due to changed channel conditions. As communication and node failures are common in wireless networks, it is possible that even these retransmitted messages do not get any response even after many retransmissions. In these cases, although the initial neighbor discovery found them to be within a one-hop distance, their communication links may fail during the execution of DRAND. This can cause deadlock. To handle these situations, we allow nodes to “give up” after some number of retransmissions. Since only *requests* and *grants* require any response, when a node does not receive any response to these messages from a one-hop neighbor for a fixed number of retries, then it removes the neighbor from its neighbor list. This allows the node to make progress with a response from the removed neighbor. This strategy can also be applied to asymmetric links. Consider a situation where a node *A* considers another node *B* as a one-hop neighbor, but *B* does not. While *A* keeps retransmitting *requests* to *B*, *B* cannot respond to *A*. In this case, *A* will eventually make progress by dropping *B* from its neighbor list.

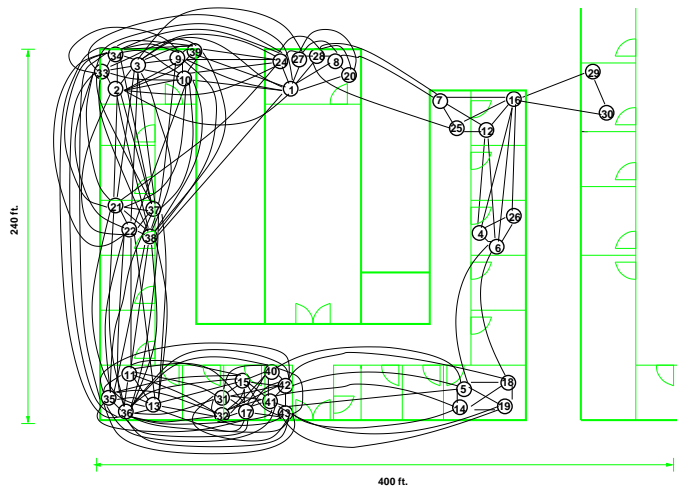
One may argue that although two nodes may not communicate in a stable manner, they might still be in an interference range so that TDMA must take care of the interference. Although our definition of conflict does not capture this situation, this interference irregularity can occur even among nodes that cannot communicate with each other at all [25]. When two nodes cannot communicate directly, their interference relations can only be deduced through multi-hop communications. There are also cases where even multi-hop communication between two interfering nodes is not possible. This issue reveals the fundamental limitations of TDMA (in a distributed manner). However, we claim that slot schedules that remove interferences only among those within a two-hop communication range are still useful in enhancing the performance of MAC. Z-MAC [21] is one example of a hybrid MAC protocol that uses such a schedule to enhance the performance of CSMA-based MAC. In cases where interference relations cannot be resolved by the schedule, Z-MAC behaves like CSMA. This contention resolution technique allows TDMA schedules to be used safely without degrading its performance too much in ad hoc networks.

## 6. EXPERIMENTAL RESULTS

In this section we have the following goals: (1) we verify the analysis in Section 4, and in particular, we validate our claim that the performance of DRAND is linearly proportional to the number of one and two-hop neighbors. (2) we evaluate the overhead of DRAND when run on a medium-size Mica2 testbed consisting of 42 nodes, and (3) evaluate the effectiveness of the TDMA schedule generated by DRAND when compared to existing schemes like FPRP [27] and SEEDEX [22].

### 6.1 Experimental Setup

Although DRAND is applicable for networks other than sensor networks, we use Mica2 for the experimental platform only because it is convenient to modify the MAC functionality in software. For the other types of radios (e.g., IEEE 802.11), we use NS [5] simulation. In our TinyOS [12] implementation, we use the default setting of B-MAC [19] (CCA is on, LPL is off, and acknowledgment is disabled) and no prior clock synchronization. We use the following testing scenarios for experiments:



**Figure 4: The wireless sensor network testbed topology. 42 Mica2 sensors are placed over two buildings.**

**One-Hop Mica2 Experiments:** A varying number of Mica2 wireless sensors are placed within a one-hop neighborhood. We vary the number of nodes in the network from one node to twenty nodes. All nodes are placed at least 2 feet above the ground.

**Two-Hop and Multi-Hop Mica2 Experiments:** Figure 4 shows our multi-hop wireless testbed consisting of 42 Mica2 sensors deployed in offices and classrooms of our computer science building. The radio connectivity between two nodes (shown by a line connecting them) varies in quality, with some links having loss rates as high as 30-40%. Our two-hop topology consists of two clusters of 10 nodes each and a sink node (node 36) selected from the nodes in this topology. The two clusters can only communicate directly with the sink node, and thus act as *hidden terminals* for each other.

**Multi-Hop NS Experiments:** We study how DRAND scales up to large scale ad-hoc wireless networks using the Network Simulator (NS). The network topology consists of nodes placed randomly on a 300mx300m surface. Nodes have a radio range of 40m and a link capacity of 2 Mbps. The neighborhood size of the network is changed by varying the number of nodes from 50 to 250. This setup produces topologies with the average number of nodes within two hops varying between 5 and 60.

### 6.2 Validation of Analysis

**Time complexity and number of rounds:** Figure 5 (A) shows the average of the maximum number of rounds and running time that a node has taken to decide on its slot and the average of the maximum number of rounds taken by a node in the one-hop topology. Experiments are repeated 10 times. The error bars denote 95% confidence intervals. Note that in the one-hop case, all nodes are within radio range of each other, as a result, DRAND assigns a unique time slot to each node, which is also the optimal.

Our measurements show that the number of rounds follows the analysis as it grows linearly with the size of the

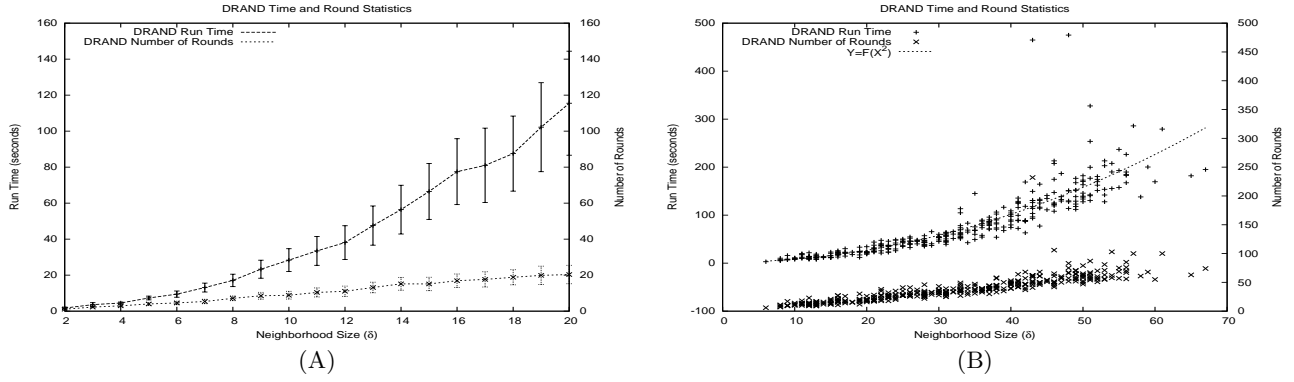


Figure 5: The average time and number of rounds taken for a node to acquire a time slot on the one-hop Mica2 (A) and multi-hop NS (B) topologies. In (B),  $Y = F(X^2)$  represents the least squares fit to the observed running time.

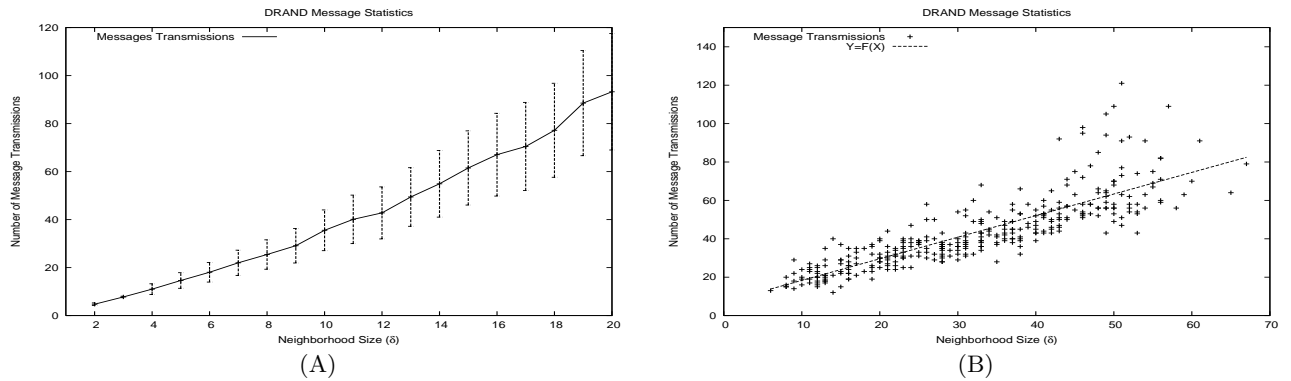


Figure 6: The average number of message transmissions per node during the execution of DRAND on the one-hop Mica2 (A) and multi-hop NS (B) topologies. In (B),  $Y = F(X)$  represents the least squares fit to the observed number of message transmissions.

neighborhood. However, the running time grows quadratically. This is because the message delays are not bounded by a constant and have a dependency to the size of the network. Thus, while the number of rounds grow linearly, the time duration of each round is not constant. Figure 5 (B) shows the same on the NS topologies. In these networks, nodes are within multiple hops to each other. Again, the run time of DRAND is quadratic with two-hop neighborhood size while the number of rounds required for each run grows linearly.

**Message Complexity:** Figures 6 (A) and (B) show the average number of message transmissions per node for the one-hop Mica2 and the multi-hop NS topologies respectively. In both cases we see that it grows linearly with the neighborhood size. Note that in the case of the NS experiment, as the neighborhood size increases, the number of message transmissions also begin to exceed the linear bound. This is because we include the retransmitted messages in the message count, and with increasing network density, the number of collisions also increases, causing an increase in the number of retransmitted messages. The effect of network contention and message losses due to collisions and noise is not captured by our model, hence the discrepancy.

**Maximum Slot Number:** Recall that in DRAND, the maximum slot number is bounded by  $\delta + 1$ . But in practice, the number of time slots that DRAND assigns can be far less than that. Figure 7 shows the number of slots used for input graphs with various densities. The dotted line indicates  $\delta$ . Each data point represents the maximum number of time slots being assigned by DRAND for different networks. In all runs, the maximum slot number used by DRAND is far less than  $\delta + 1$ . This is in contrast to the performance of other algorithms such as [4, 27, 6, 15] whose worst case performance is always larger than or equal to  $\delta + 1$ .

### 6.3 DRAND Overhead Cost

**Time and Energy Cost:** In this section we examine the energy cost of running DRAND on a medium-sized Mica2 multi-hop network described in Section 6.1. Each node records the total number of primitive radio operations (receive a byte, transmit a byte and idle listening). The total energy is calculated to be the sum of these operations weighted by the energy cost of each operation as reported in [19].

On this topology, each node first runs a neighbor discovery protocol to get its neighborhood information. Then the DRAND algorithm is executed and a TDMA time slot is assigned to each node. Finally, nodes disseminate their

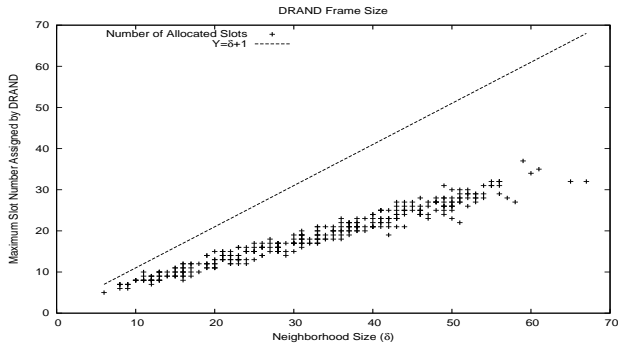


Figure 7: The maximum number of time slots being assigned by DRAND for input graphs with varying neighborhood size ( $\delta$ ). The line is  $Y = \delta + 1$ , which is the worst case upper bound for DRAND (and RAND).

slot information to their two-hop neighborhood so that data transmission may start using this slot information.

Operation	Average Time	Average Energy
Neighbor Discovery	30s	0.732J
DRAND Slot Assignment	194.38s	4.88J
Slot Dissemination	60s	1.33J
Total	284.38	6.942J

Table 1: DRAND Overhead Cost

The average costs of 30 runs of the full testbed scenario for each of the three phases – neighbor discovery, DRAND and the slot size dissemination are shown in Table 1. The energy and time reported are the average of the maximum for all nodes in the testbed in each run. Note that all three phases together take 6.942 J which is about 0.02% of the total battery capacity of a node with 2500 mAh and 3 V battery (the same battery used in Table 3 [19]).

The slots assigned to each node for one particular run are shown in Figure 8. While most nodes within two hops of each other are assigned unique time slots, a few discrepancies do exist – consider nodes 7, 12 and 30 which belong in the same two-hop neighborhood, but have been assigned the same slot, namely 0. This is due to the presence of asymmetric links between these nodes. As described in Section 5, in the presence of asymmetric links, the DRAND algorithm makes progress by dropping non-responding neighbors.

**DRAND Recovery Cost:** In DRAND, when a new node **A** joins the network (or a node, already assigned a slot, fails for some reason and restarts), it can secure a time slot by (re-)running DRAND within its one hop-neighborhood – it is not necessary for the nodes more than one hop away from **A** to re-run DRAND. We simulate node join by restarting specific nodes on our testbed, allowing all nodes within one-hop of the restarted node to run DRAND and then measure the average time and energy for all such nodes to secure new conflict-free time slots. We choose 5 such nodes – (12, 19, 15, 22 and 3). Figure 9 shows the average time and energy expended for the simultaneous restart of node sets (12), (12, 19), (12, 19, 15), (12, 19, 15, 22) and (12, 19, 15, 22, 3). When

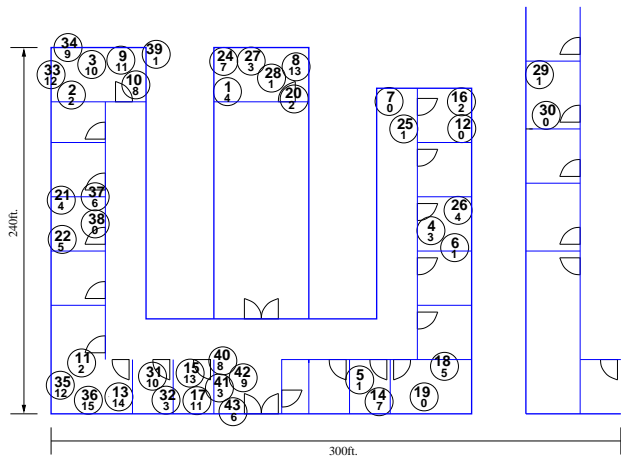


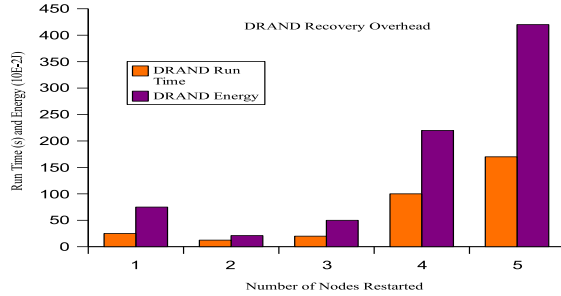
Figure 8: DRAND time slot assignment on the wireless sensor network testbed. The numbers on each node represent the node ID and the TDMA slot number assigned, respectively. The maximum slot number assigned to a node on this testbed is 15. By comparing the network connectivity shown in Figure 4, one can verify that, with the exception of a few nodes (e.g. 7,12,30), most nodes within two hops of each other are assigned different time slots. The discrepancies are due to the presence of asymmetric links.

all five nodes fail, their one-hop neighborhood together includes all nodes in the testbed, and hence the cost is approximately the same as that for re-running DRAND again for all nodes.

## 6.4 Comparison with Existing Algorithms

We now compare DRAND with the following schemes:

- (1) **FPRP:** FPRP [27], Five Phase Reservation Protocol, is a distributed heuristic TDMA slot assignment algorithm. FPRP is designed for dynamic slot assignment, in which the real time is divided into a series of a pair of *reservation* and *data transmission* phases. During the reservation phase, the protocol assigns the slots of the next data transmission phase to the nodes who have data to send. Thus, the actual data transmission occurs using the slots assigned in the previous reservation phase. For each time slot of the data transmission phase, FPRP runs a five-phase protocol for a number of times (cycles) to pick a winner of each slot. The algorithm improves its assignment as it runs for more cycles. In a distributed setting, nodes do not know how many cycles are required to complete the assignment. Hence, for the sake of comparison, we measure the number of slots assigned by FPRP by fixing the number of cycles to run.
- (2) **Randomized TDMA (R-TDMA):** In R-TDMA, the network maintains a fixed frame size  $F$ , where  $F$  is set to be equal to  $\delta$ , the maximum size of a two-hop neighborhood in the network. At the beginning of each frame, a node with a packet ready for transmission chooses a slot  $i$  between 0 and  $F - 1$  with uniform probability. At slot  $i$ , the node transmits the packet.



**Figure 9: The average time and energy required for a node to execute DRAND and (re-)join a network. To simulate this condition, node sets of size 1, 2, 3, 4 and 5 nodes from different parts of the wireless testbed are restarted. This causes all nodes within one-hop of the restarted node to re-execute DRAND. The measured time and energy are shown. When five nodes are restarted, their one-hop neighborhood includes all nodes in the testbed, and hence the total time and energy is close to re-running DRAND for the whole testbed.**

- (3) **SEEDEx:** In SEEDEx [22], at the beginning of each slot, if a node has a packet ready for transmission, it draws a “lottery” with probability  $p$ . If it wins, it becomes *eligible* to transmit. A node knows the seeds of the random number generators of its two-hop neighbors, and hence it also knows the number of nodes (including itself), within two hops ( $C$ ) who are also eligible to transmit. It then transmits with probability  $1/C$ . [4] also uses random seeds of neighboring nodes to determine slots. This technique is also called *topology independent scheduling*.

**Algorithm Complexity:** Figure 10 shows the performance results of DRAND and FPRP with respect to the maximum slot number, the number of transmitted messages per node and the total run time of each algorithm. The experiment is conducted on the multi-hop NS topologies, as we vary the average number of two-hop neighbors from 8 to 52. Note that SEEDEx and R-TDMA do not have the concept of the maximum slot number. Both SEEDEx and R-TDMA run continuously, and hence we do not perform these comparisons for these protocols – instead we look at their transmission efficiency later.

FPRP- $x$  indicates the performance of FPRP when we run  $x$  cycles of FPRP to find the winner of each time slot. As the neighborhood size increases, there is a corresponding increase in the number of slots, the message complexity and the running time of FPRP. DRAND outperforms all of FPRPs on the number of slots and message complexity. The number of slots required by DRAND is always less than that of FPRP (up to 34%). Note that any percentage reduction in the number of slots can be translated into the same percentage improvement of the overall performance in channel utilization since it represents the size of frames in TDMA. The running time of DRAND is comparable to that of FPRP-30, and is less than FPRP-50. Considering that DRAND produces much more efficient slot assignments than FPRP-50, this result is encouraging. The number of trans-

mitted messages in DRAND is far less than that in FPRP, which implies much less energy consumption for DRAND.

**Transmission Efficiency:** We now examine how the TDMA slot assignments generated by DRAND, SEEDEx and R-TDMA affect the throughput when used in a TDMA MAC protocol. (Note that FPRP has already been shown to generate more slots than DRAND, and hence we do not evaluate its performance in this regard.) We implemented a vanilla TDMA MAC scheme in TinyOS in order to conduct the above experiment in Mica2 sensor nodes. A node sends a packet *only* during its own assigned slot. We compare the cases where slots are chosen by DRAND, SEEDEx and R-TDMA respectively. The tests are run in our testbed shown in Figure 4.

Before the beginning of data transmission, we first ensure that all nodes are synchronized with respect to their TDMA slots, using a localized time synchronization scheme described in [21]. In spite of time synchronization, nodes may lose synchronization due to the loss of time synchronization messages and clock drift between synchronizations. Hence, we enforce nodes not to transmit during 5ms of the beginning of each time slot – 5ms is used as a slack time for synchronization error. This slack time prevents collisions occurring at the boundary of time slots due to time synchronization errors. Note that the Mica2 radio (CC1000 [1]) takes about 19.2ms to transmit a packet of size 46 bytes (10 bytes for the MAC preamble and sync, and 36 bytes for the payload) under the radio data rate of 19200 bytes per second. Hence, we set TDMA time slots to a period of 25ms, which is sufficiently long for the transmission of one packet and the slack time.

We run the experiment on both one and two hop Mica2 topologies picked from part of the network in the testbed. For all three protocols, the one hop results were only slightly better than the two hop results (due to the absence of hidden terminals), and hence we do not report them. For SEEDEx, we run the experiment with  $p = 0.246, 0.117$  and  $0.074$ , which are optimal for neighborhood sizes of 6, 12 and 21, respectively, according to the analysis in Section 7 of [22]. Figure 11 (A) shows the throughput obtained at the sink as we increase the number of senders. We do not show results for  $p = 0.246$  and  $0.117$ , since the performance is not better than  $p = 0.074$ . We find that deterministic TDMA schemes like DRAND always give better performance compared to randomized TDMA schemes like SEEDEx and R-TDMA. This is due to two reasons. (1) The randomized nature of the schemes results in more collisions (up to 20% more), as we observe from Figure 11 (B). (2) Randomized schemes also tend to transmit lesser number of packets on average, as seen in Figure 11 (C).

Of course, it is possible to increase the number of transmissions for randomized schemes, by changing particular parameters of the scheme. For instance, by reducing  $p$  below the optimal value of  $0.074$ , one can expect the number of transmissions to exceed that of DRAND, but at the same time this would cause an increase in the collisions, hence reducing throughput. By using the optimal parameter of  $p$ , we ensure that we are sending at the particular transmission rate under SEEDEx that gives the best throughput.

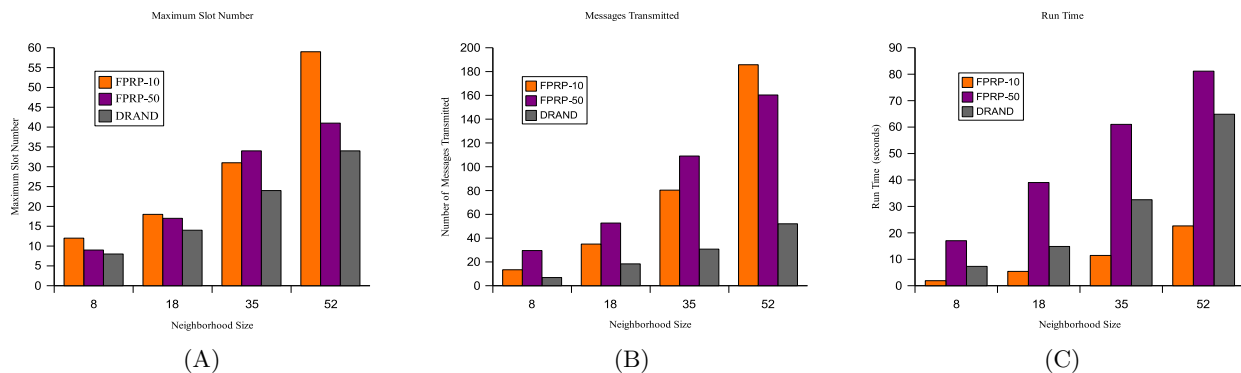


Figure 10: The maximum slot number, average number of message transmissions, and average time required in FPRP and DRAND for a node to acquire a time slot. The experiment is conducted on the multi-hop NS topologies varying the average number of two-hop neighbors from 8 to 52. FPRP- $x$  indicates that we run  $x$  FPRP cycles to find the winner for each time slot.

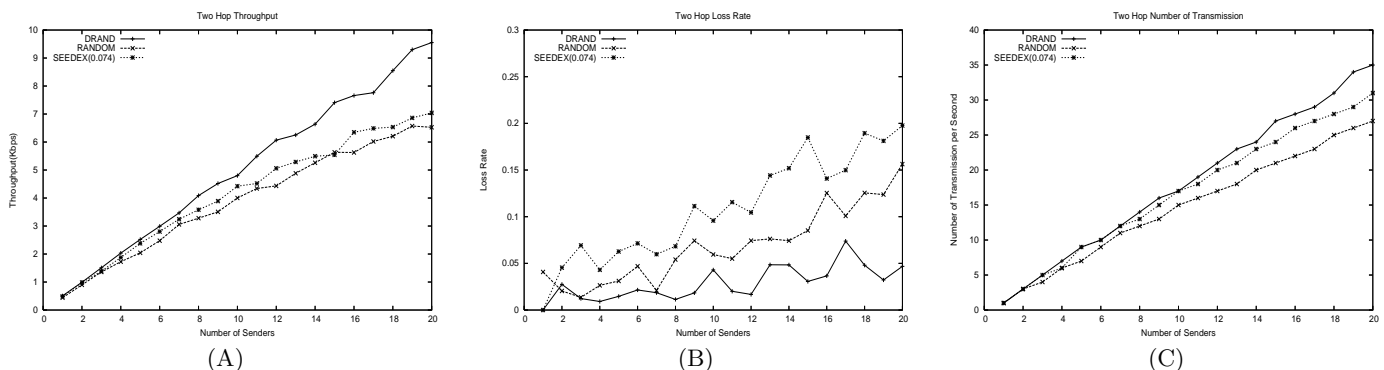


Figure 11: The throughput (A), loss rate (B) and number of transmissions per node (C) for a two-hop Mica2 topology running a TDMA protocol using R-TDMA, SEEDEX and DRAND. Two clusters of 10 nodes each are selected from the wireless testbed shown earlier. The number of senders is varied from 1 to 20 by picking one node from each cluster alternately for each run. All nodes always have a packet ready for transmission.

## 7. RELATED WORK

Most of the early work in TDMA scheduling is centralized and has performance dependency to  $O(n)$  where  $n$  is the total size of the network. Some distributed solutions [4, 27, 26, 22, 6] improve the performance by removing dependency on the global topology. These algorithms are developed for mobile environments where nodes can frequently move and typically use many more time slots than  $\delta + 1$  (for some protocols, e.g., [6, 27], these bounds are not given).

NAMA [4] uses a hash function to determine priority among contending neighbors. One main drawback of this hashing based technique is priority chaining; even though a node gets a higher priority in one neighborhood, it may still have a lower priority in other neighborhoods. This chaining can build up to  $O(n)$ , yielding a sub-optimal schedule. Thus the maximum slot number of NAMA is  $O(n)$ . In FPRP [27] and E-TDMA [26] nodes select slots randomly using a five-phase algorithm. But it is possible that a node may not be assigned to a slot and requires many runs to increase the chance that a node gets assigned to a slot. SEEDEX [22] uses a similar hashing scheme as NAMA based on a random seed exchanged in a two-hop neighborhood. Collisions may still occur if two nodes select the same slot and decide to transmit.

Kulkarni et al. [14] propose a solution where each node runs a local coloring algorithm, and they do so in the order that they receive a token which is passed around over a minimum spanning tree created by a base station. The coloring obtained is  $O(\delta^2)$ . In NB-TDMA [18], TDMA scheduling is done on demand. A node wishing to transmit data towards a sink, dispatches a *mobile agent*, which traverses each node on the routing path to the sink, creating a schedule in the process. This creates a coupling between the routing and the MAC operation, and hence routing changes would necessitate changes in the schedule. Also, short-lived flows could end up facing long delays for the schedule to be set up.

Moscibroda et al. [16] propose a graph coloring scheme with a time complexity of  $O(\rho \log n)$ , where  $\rho$  is the maximum node degree in the network. The scheme performs *distance-1 coloring*, in that adjacent nodes have different colors. Note that this does not prevent nodes within two hops of each other from being assigned the same color – potentially causing hidden terminal collisions between such nodes. DRAND performs distance-2 coloring, hence nodes within two hops are assigned different colors. Parthasarathy et al. [17] propose a distance-2 coloring algorithm similar to DRAND, which uses  $O(\delta)$  colors and has time complexity  $O(\delta \log^2 n)$  and message complexity of  $O(n \log^2 n)$ . The al-

gorithm proceeds in rounds and requires all nodes to be synchronized on the round boundary. The DRAND algorithm also operates in rounds, but the nodes are not required to be synchronized on the round boundary. Herman et al. [11] propose a distributed TDMA slot assignment algorithm based on a distance-2 coloring scheme. The algorithm first finds a conflict-free assignment of colors, and then tries to allocate time slots to these colors, while maintaining fairness within a two-hop neighborhood. The algorithm requires each node to maintain state within its three-hop neighborhood, which could be quite difficult and resource intensive. In comparison, in DRAND, a node only needs to maintain state within its one-hop neighborhood. In addition, nodes are assumed to be synchronized with respect to a global clock. Again, DRAND does not require time synchronization.

A different, but related problem to TDMA node slot assignment, is the problem of TDMA edge slot assignment, where radio links (or edges) are assigned time slots, instead of nodes. Finding the minimum number of time slots for a conflict free edge slot assignment is an NP-complete problem [20]. Edge slot assignment may lead to better spatial reuse [9] than node slot assignment. But it requires each node to be aware of predetermined routing paths. In ad hoc networks, with time varying channels, these paths can change frequently. Thus, link scheduling in general may not be so adaptive to these changes. Salonidis et al. [23] propose an algorithm for dynamic link scheduling of tree routing structures commonly found in ad-hoc networks. Gandham et al. [8] present a distributed algorithm which assigns slots to edges of acyclic graphs with at most  $2(\rho + 1)$  time slots, where  $\rho$  is the maximum degree of the graph. For arbitrary graphs, their experimental results show that the number of required time slots could go up to  $4\rho$ . Note that DRAND can perform slot assignment for arbitrary graphs. Authors do not provide asymptotic performance bounds on time and message complexity of their algorithm.

## 8. CONCLUSION

We introduce DRAND, a distributed, robust and scalable implementation of RAND, a centralized TDMA scheduling algorithm for wireless ad-hoc networks. DRAND is ideal for wireless networks with limited mobility, e.g., wireless mesh networks and wireless sensor networks. DRAND can accommodate some degree of topology changes by performing localized operations. We implement DRAND in the resource-limited environment of wireless sensor networks (Mica2 sensors running TinyOS) and show that DRAND performs robustly in real wireless settings. Compared to existing schemes, DRAND gives more efficient slot assignments which result in better channel utilization while at the same using less network resources and energy.

## 9. REFERENCES

- [1] CC1000 Low Power FSK transceiver, Chipcon Corporation.
- [2] WiMedia Alliance, MBOA Wireless MAC Specification for High Rate Wireless Personal Area Networks (WPANs), Specification Draft.
- [3] ZigBee Alliance, IEEE 802.15.4, ZigBee standard.
- [4] L. Bao and J. J. Garcia-Luna-Aceves. A new approach to channel access scheduling for ad hoc networks. In *ACM MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 210–221, New York, NY, USA, 2001. ACM Press.
- [5] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu. Advances in network simulation. *IEEE Computer*, 33(5):59–67, May 2000.
- [6] I. Chlamtac and A. Farag. Making transmission schedules immune to topology changes in multi-hop packet radio networks. *IEEE/ACM Transactions on Networking*, 2(1):23–29, 1994.
- [7] B. Crow, I. Widjaja, J. G. Kim, and P. Sakai. IEEE 802.11 wireless local area networks. *IEEE Communications Magazine*, 35(9):116–126, 1997.
- [8] S. Gandham, M. Dawande, and R. Prakash. Link scheduling in sensor networks: Distributed edge coloring revisited. In *IEEE INFOCOM*, 2005.
- [9] J. Grnkvist. Assignment methods for spatial reuse TDMA. In *ACM MobiHoc '00: Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, pages 119–124, Piscataway, NJ, USA, 2000. IEEE Press.
- [10] M. Gruteser and D. Grunwald. Enhancing location privacy in wireless lan through disposable interface identifiers: a quantitative analysis. In *WMASH '03: Proceedings of the 1st ACM international workshop on Wireless mobile applications and services on WLAN hotspots*, pages 46–55, New York, NY, USA, 2003. ACM Press.
- [11] T. Herman and S. Tixeuil. A distributed TDMA slot assignment algorithm for wireless sensor networks. In *Proceedings of the First Workshop on Algorithmic Aspects of Wireless Sensor Networks (AlgoSensors'2004)*, number 3121 in Lecture Notes in Computer Science, pages 45–58, Turku, Finland, July 2004. Springer-Verlag.
- [12] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for network sensors. In *Proc. Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Cambridge, MA, November 2000.
- [13] B. Hull, K. Jamieson, and H. Balakrishnan. Mitigating congestion in wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 134–147, New York, NY, USA, 2004. ACM Press.
- [14] S. S. Kulkarni and M. U. Arumugam. TDMA service for sensor networks. In *ICDCSW '04: Proceedings of the 24th International Conference on Distributed Computing Systems Workshops - W7: EC (ICDCSW'04)*, pages 604–609, Washington, DC, USA, 2004. IEEE Computer Society.
- [15] M. Luby. Removing randomness in parallel computation without processor penalty. *Journal of Computer and System Sciences*, 47(2):250–286, Oct. 1993.
- [16] T. Moscibroda and R. Wattenhofer. Coloring Unstructured Radio Networks. In *17th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), Las Vegas, Nevada, USA*, July 2005.
- [17] S. Parthasarathy and R. Gandhi. Distributed

- algorithms for coloring and domination in wireless ad hoc networks. In *FSTTCS*, pages 447–459, 2004.
- [18] R. K. Patro and B. Mohan. Mobile agent based TDMA slot assignment algorithm for wireless sensor networks. In *International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II*, pages pp. 663–667, 2005.
- [19] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *ACM SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107, New York, NY, USA, 2004. ACM Press.
- [20] S. Ramanathan. A unified framework and algorithms for (T/F/C)DMA channel assignment in wireless networks. In *IEEE INFOCOM*, pages 900–907, 1997.
- [21] I. Rhee, A. Warriar, M. Aia, and J. Min. Z-MAC: a hybrid MAC for wireless sensor networks. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 90–101, New York, NY, USA, 2005. ACM Press.
- [22] R. Rozovsky and P. R. Kumar. SEEDEX: a MAC protocol for ad hoc networks. In *ACM MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 67–75, New York, NY, USA, 2001. ACM Press.
- [23] T. Salonidis and L. Tassiulas. Distributed dynamic scheduling for end-to-end rate guarantees in wireless ad hoc networks. In *ACM MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 145–156, New York, NY, USA, 2005. ACM Press.
- [24] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the IEEE Infocom*, pages 1567–1576, New York, NY, USA, June 2002. USC/Information Sciences Institute, IEEE.
- [25] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic. Impact of radio irregularity on wireless sensor networks. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 125–138, New York, NY, USA, 2004. ACM Press.
- [26] C. Zhu and M. Corson. An evolutionary-TDMA scheduling protocol (E-TDMA) for mobile ad hoc networks. In *Proc. of Advanced Telecommunications and Information Distribution Research Program (ATIRP)*, March, 2000.
- [27] C. Zhu and M. S. Corson. A five-phase reservation protocol (fprp) for mobile ad hoc networks. *Wireless Networks*, 7(4):371–384, 2001.