Consensus and Collision Detectors in Wireless Ad Hoc Networks

Gregory Chockler
grishac@csail.mit.edu

Murat Demirbas
demirbas@mit.edu

Seth Gilbert
sethg@mit.edu

Calvin Newport
cnewport@mit.edu

Tina Nolte
tnolte@mit.edu

MIT Computer Science and Artificial Intelligence Lab
Cambridge, MA 02139, USA

ABSTRACT

We consider the fault-tolerant consensus problem in wireless ad hoc networks with crash-prone nodes. We develop consensus algorithms for single-hop environments where the nodes are located within broadcast range of each other. Our algorithms tolerate highly unpredictable wireless communication, in which messages may be lost due to collisions, electromagnetic interference, or other anomalies. Accordingly, each node may receive a different set of messages in the same round. In order to minimize collisions, we design *adaptive* algorithms that attempt to minimize the broadcast contention. To cope with unreliable communication, we augment the nodes with *collision detectors* and present a new classification of collision detectors in terms of accuracy and completeness, based on practical realities. We show exactly in which cases consensus can be solved, and thus determine the requirements for a useful collision detector.

We validate the feasibility of our algorithms, and the underlying wireless model, with simulations based on a realistic 802.11 MAC layer implementation and a detailed radio propagation model. We analyze the performance of our algorithms under varying sizes and densities of deployment and varying MAC layer parameters. We use our single-hop consensus algorithms as the basis for solving consensus in a multi-hop network, demonstrating the resilience of our algorithms to a challenging and noisy environment.

*This work is supported by MURI–AFOSR SA2796PO 1-0000243658, USAF–AFRL #FA9550-04-1-0121, NSF Grant CCR-0121277, NSF-Texas Engineering Experiment Station Grant 64961-CS, and DARPA F33615-01-C-1896.

PODC'05, July 17–20, 2005, Las Vegas, Nevada, USA.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms

Theory, Algorithms, Reliability

Keywords

Wireless ad hoc networks, sensor networks, collision detection, consensus, fault-tolerance

1. INTRODUCTION

As wireless technology has improved and miniaturized, there has been an increasing interest in large-scale, widely-deployed sensor networks. Many service and applications in these environments (e.g., TDMA scheduling, remote management and re-programming of sensors, temperature and climate control, assembly line monitoring, etc.) require wireless devices to coordinate their actions in the face of failures resulting from hardware malfunction, physical damage, battery depletion, or enforced hibernation. Fault-tolerant agreement (or *consensus*) is a quintessential building block for these applications as it facilitates maintenance of consistent replicated state on which the participants can act in a consistent manner.

In this paper, we study the fault-tolerant consensus problem in wireless ad hoc networks with crash-prone nodes. For most of the paper, we focus on solving consensus in *single-hop* networks where the nodes are located within communication range of each other and are tightly synchronized. Due to these assumptions, consensus might appear to be trivially solvable. However, as we discuss below, real wireless networks pose several additional difficulties that rule out the trivial solutions.

First, communication in wireless networks is unreliable: collisions and other wireless interference might cause significant message disruption. Second, the deployment of devices cannot be carefully controlled, so the number of deployed devices (and, perhaps, the density of the deployment) is *a priori* unknown. Moreover, the devices may be "anonymous",

meaning that they have no unique identifiers. As a result of collisions, an arbitrary number of messages that have been broadcast in a round can be lost. Furthermore, without an *a priori* knowledge of the number of participants, the message loss cannot be reliably detected.

To circumvent the problem of unrestricted message loss, we assume that, *eventually*, if the broadcast contention is low enough in a given round, then the MAC layer is able to ensure that there are no collisions. The fact that this property only holds eventually prevents nodes from simply assuming that a broadcast was successfully received based only on a known number of concurrent broadcasters.

Under this assumption, we focus on developing *adaptive* algorithms where the number of broadcasting participants is dynamically adjusted toward the collision-free contention level, without actually knowing the value of this threshold or the number of participants. The two main advantages of this approach are (1) improved fault-tolerance for MAC layers that can sustain higher contention levels, and (2) the ability to use fixed length rounds, which is important in practice.

Collision Detectors

To cope with undetectable message loss, we augment the nodes with *collision detection*. Collision detectors monitor the broadcast medium and attempt to deliver notifications when message loss is detected. They do not provide any information with respect to the number of lost messages or the identities of their senders. Moreover, there is no guarantee that a node performing a transmission can detect collisions (unlike, for example, Ethernet networks [29]).

Inspired by [7], we classify collision detectors according to their *completeness*, the ability to detect actual collisions, and *accuracy*, the ability to report only actual collisions (no false positives). For each collision-detector class that we introduce, we show how to solve consensus and provide matching lower bounds.

We consider two accuracy properties: permanent accuracy and eventual accuracy. While always accurate collision detectors are more powerful, eventually accurate collision detectors are more realistic, since they result in algorithms that are robust in the face of false positives caused by electromagnetic noise and broadcasts by nearby nodes. The latter is particularly important for multi-hop algorithms that use single-hop consensus as a building block: In these algorithms, neighboring instantiations of single-hop consensus can interfere with each other, leading to false collision detection.

Since most current collision detector implementations can occasionally miss a collision, we also consider two ways of weakening the assumption of completeness. In particular, we consider: (1) a *majority complete* collision detector that only detects a collision if a *majority* of messages in a round is lost, and (2) a *0-complete* collision detector that only detects a collision if *every* message in a round is lost.

We consider the six collision detector classes obtained by combining the completeness and accuracy properties above (see Table 1). We analyze the computational power of each of these classes in terms of the following parameters: (1) an ability to solve consensus, (2) the solution complexity, and (3) robustness to message loss (see Table 2). Our results provide a separation among all of these classes in terms of the parameters above. An important contribution of our analysis is in providing feedback to hardware/firmware designers with respect to the requirements for collision detectors. While there recently has been significant progress in implementing collision detection [13, 34, 43], there has been little formal analysis of the minimal requirements. We show that reasonable and readily implementable collision detectors are sufficient.

Experimental Results

We demonstrate the utility of our single-hop consensus primitive by using it to develop a simple and efficient multi-hop consensus protocol. In this protocol, the multi-hop network is divided into a series of non-overlapping grid squares, and each node knows its approximate location in the grid. We use single-hop consensus within each grid square to reach a local decision which is then propagated to the other grid squares. This reduces the sensitivity to varying deployment densities by effectively aggregating the initial values. It therefore both reduces bandwidth for the propagation phase of the algorithm, and compartmentalizes much of the complexity of consensus.

We implemented our single and multi-hop algorithms in a simulator featuring a detailed radio propagation model and a realistic MAC layer implementation. We used the simulated implementation to thoroughly analyze the performance of the algorithms and assess the significance of various parameters (such as the number of participating nodes, the round length, the low-level collision-avoidance scheme, etc.) on the their efficiency. The evaluation results are encouraging and validate our claim that our algorithms are adaptive to varying densities and varying levels of communication contention. They show that our single-hop consensus protocol sustains up to 100 nodes with only a marginal increase in the number of rounds required to reach consensus and the latency of the multi-hop algorithm is minimally affected by the increase in the node density.

2. RELATED WORK

There has been extensive prior research on fault-tolerant consensus in synchronous (see [27]), partially synchronous (e.g., [14]), asynchronous with failure detectors (e.g., [7, 24]) and fully asynchronous (e.g., [16]) message passing systems with reliable or eventually reliable point-to-point channels. In particular, [14, 24] overcome message loss by assuming that eventually there is a connected majority component. This assumption is unavailable in the wireless ad hoc environments we consider.

Santoro and Widmayer [35, 36] study consensus in the presence of unreliable communication, and show that consensus is impossible if as few as $(n-1)$ of the n^2 possible messages sent in a round can be lost. In this paper, we circumvent this impossibility result by exploiting collision-detection information. Also, algorithms in [36] are not applicable in our setting since they rely on a priori known number of participants, and do not tolerate node failures.

Aspnes et al. [4] present a solution for consensus in wireless networks with anonymous but reliable nodes, and reliable communication. Although anonymity is not a primary focus of our paper, most of our algorithms are, in fact, anonymous as they do not use node identifiers. In addition, our algorithms work under more realistic environment assumptions as they tolerate unreliable communication and node crashes.

	Complete	**maj-Complete**	**0-Complete**
Accurate	$\mathcal{AC}$	maj-$\mathcal{AC}$	0-$\mathcal{AC}$
Eventually Accurate	$\diamond\mathcal{AC}$	maj-$\diamond\mathcal{AC}$	0-$\diamond\mathcal{AC}$

Table 1: A summary of collision detector classes.

Koo [21] presents a tight lower bound for the minimum fraction of Byzantine neighbors that allows atomic broadcast to be solved in radio networks where each node adheres to a pre-defined transmission schedule. This result is potentially relevant to our multi-hop consensus protocols, although we do not consider Byzantine failures and assume unreliable broadcast.

While the problem of consensus has only recently been studied in wireless ad hoc networks, there is a long history of work on the reliable broadcast problem, which can potentially be used as a building block for solving consensus. A number of early papers (e.g., [19, 38, 41]) study the problem in Ethernet [17, 29] networks, where nodes can reliably detect collisions when messages are lost. Moreover, it is assumed that a transmitter can always detect whether its message was received successfully. In contrast, in wireless networks, messages can be overwhelmed by a stronger transmission signal, thus leading to undetectable collisions, and a transmitting node has no way of determining whether its message arrived successfully.

Starting with a seminal paper by Bar Yehuda et al. [6], and followed by many others (e.g., [5, 8, 23]), reliable broadcast was studied in synchronous radio networks where a node is guaranteed to deliver a message in a given time slot if and only if exactly one of its neighbors is transmitting a message in this slot. In contrast to this model, we allow for unpredictable collision patterns which in particular, might result in non-uniform message loss. Such non-deterministic behavior is frequently observed in real networks [22, 43, 45], and in fact arises in our simulations. We also do not assume any advance knowledge of a node's neighbors and therefore, cannot attribute lost messages to specific nodes in the networks. A variety of other variants to the reliable broadcast problem in a model similar to that of [6] have been considered in [3, 9, 12, 20].

We now briefly discuss the current state-of-the art in wireless network technology that motivates our environmental assumptions. First, it is well-known that wireless broadcast networks are inherently unreliable. Several recent experimental studies [18, 22, 42, 45] suggest that even with sophisticated collision avoidance mechanisms (e.g., 802.11 [1], B-MAC [34], S-MAC [44], and T-MAC [39]), and even assuming low traffic loads, the fraction of messages being lost can be as high as $20 - 50\%$.

The algorithms in this paper rely on collision detectors to overcome uncertainties in message loss. The importance and practicality of having collision information available to applications was argued in [43]. Several existing MAC layers, such as B-MAC [34], already support some collision detection capability. Moreover, the recent study by Deng et al. [13] suggests that there is no technical obstacle to adding collision detection support to the current 802.11 protocol. Although implementing perfectly complete collision detection still appears challenging, the weaker requirement of "majority completeness" appears feasible with today's hardware/firmware, since most of the undetectable message loss occurs when pairs of messages overlap. In fact, almost all currently implemented collision detectors appear to meet the requirements of "0-completeness," the weakest collision detector considered in this paper.

3. THE SYSTEM MODEL

We consider a single-hop wireless broadcast network consisting of fixed but *a priori* unknown collection of nodes $P = \{p_1, p_2, \dots\}$ where all nodes are located within communication range of each other. The number of nodes is *a priori* unknown, and nodes do not have unique identifiers.

Nodes communicate by broadcasting messages. A node p_i broadcasts messages by invoking $\mathsf{bcast}(m)_i$, where m is an arbitrary message, and receives messages by invoking $\mathsf{recv}()_i$. We assume that the system is synchronous: both the nodes' clock skews and the inter-node communication delay are bounded by known constants. For simplicity, we assume that the processing is divided into synchronous *rounds*. In each round r, each node p_i executes the following steps: (1) broadcasts at most one message, (2) receives a subset of messages that were broadcast by the nodes in P in round r, and (3) performs a state transition based on its current state and the set of received messages.

Nodes can fail by crashing at any point during the execution of the algorithm. However, nodes cannot crash in the middle of executing the bcast instruction. A node that does not crash throughout an entire run is said to be *correct*. Otherwise, it is said to be *faulty*.

The broadcast communication within each round satisfies the basic integrity and no-duplication properties guaranteeing that every received message was previously broadcast, and that each message is received at most once. The communication medium is prone to *collisions*. As a result of a collision, a node can loose an arbitrary subset of messages that have been broadcast in a round. Moreover, collisions may affect nodes in a non-uniform way: For example, when a node broadcasts a message, some nodes may receive it while others may not.

Since some degree of reliable message delivery is a prerequisite for many applications (and in particular, for consensus), it is commonly assumed that the underlying communication layer supports collision-free communication when transmissions do not overlap (see e.g., [5, 6, 8, 23]). In practice, however, existing wireless MAC layers often employ best-effort protocols (such as exponential back-off) that support collision-free communication even if multiple nodes simultaneously broadcast messages. We model this as follows:

Property 1 (Eventual Collision Freedom). *There exists a positive integer b, such that in each execution, there exists a round r_{ecf} so that the following is satisfied: For each round $r \geq r_{ecf}$, if at most b nodes broadcast messages in r, then all correct nodes receive all the messages that have been broadcast in r.*

Note that this property implies the following property assumed in prior work: namely, for each round $r \geq r_{ecf}$, if

only one node broadcasts in r, then every message is reliably delivered.

In order to take advantage of Eventual Collision Freedom, our algorithms use a special type of contention-management mechanism, called a *wake-up service*, that determines which nodes should broadcast in a given round. A contention manager is a service that can be queried in each round to determine whether a node should be *active* or *passive* in the round. We say that a contention manager provides *good advice* if it recommends that at least one, and no more than b, correct nodes are active, where b is the unknown parameter whose existence is posited by Eventual Collision Freedom. A contention manager is called a *wake-up* service if it guarantees to eventually provide good advice. Formally:

Property 2. *There exists a round r_{wake} such that for each $r \geq r_{wake}$, the wake-up service provides good advice in round r.*

A wake-up service can be implemented using a randomized back-off protocol, as the one outlined in Section 8.

4. COLLISION DETECTORS

As we prove elsewhere [10], consensus is impossible in collision-prone environments, even with Eventual Collision Freedom, if the number of participants is *a priori* unknown. We therefore assume that the MAC layer of every node $p_i \in P$ is augmented with a *collision detector*. A node p_i learns about a possible collision in round r when the set of messages received in round r includes a *collision notification* $\pm$. In this case, we say that p_i *detects* a collision in round r. Note that collision notifications only indicate a possible message loss in a round. In particular, they do not provide any information with respect to the number of lost messages, and the identities of their senders.

Inspired by the way in which [7] presents failure detectors, we classify collision detectors in terms of the completeness and accuracy properties satisfied by each collision detector in the class. A collision detector satisfies *completeness* if the following holds:

Completeness: For every round r of each execution, if p_i does not receive some messages that were broadcast in r, then p_i detects a collision in r.

A collision detector satisfies *accuracy* if the following holds:

Accuracy: For each round r of every execution and a node $p_i \in P$, if p_i detects a collision in r, then p_i does not receive some messages that were broadcast in r.

As we discuss in the introduction, in many practical scenarios, the MAC layer can reliably detect collisions only if a certain fraction of the messages being broadcast in a round is lost. To this end, we consider collision detectors satisfying the following: Let $M(r)$ denote the number of bcast events that occur in round r. A collision detector satisfies *majority completeness (maj-Completeness)* if the following holds:

maj-Complenetess: For each round r of every execution and a node $p_i \in P$: If p_i receives $\leq M(r)/2$ messages in r, then p_i detects a collision in r.

A collision detector satisfies 0-Completeness if the following holds:

0-Completeness: For each round r of every execution in which $M(r) > 0$ and a node $p_i \in P$: If p_i does not receive any messages in r, then p_i detects a collision in r.

Finally, in order to account for situation in which arbitrary noise can be mistaken for collisions, we will consider collision detectors satisfying the following property:

Eventual Accuracy: For each execution, there exists a round r_{acc} such that for each round $r' \geq r_{acc}$, and each process $p_i \in P$: If p_i detects a collision in r', then p_i does not receive some messages that were broadcast in r'.

For the sake of the presentation, we will refer to collision detectors satisfying completeness as *reliable*, and to those satisfying either variant of weak comleteness as *unreliable*. The collision detectors considered in this paper are summarized in Table 1.

	Eventual Collision Freedom	**No Collision Freedom**
$\mathcal{AC}$	$\Theta(1)$	$\Theta(\log \lvert V \rvert)$
maj-$\mathcal{AC}$	$\Theta(1)$	$\Theta(\log \lvert V \rvert)$
0-$\mathcal{AC}$	$\Theta(\log \lvert V \rvert)$	$\Theta(\log \lvert V \rvert)$
$\diamond\mathcal{AC}$	$\Theta(1)$	Impossible
maj-$\diamond\mathcal{AC}$	$\Theta(1)$	Impossible
0-$\diamond\mathcal{AC}$	$\Theta(\log \lvert V \rvert)$	Impossible

Table 2: Solving consensus with different collision detector classes. In Sections 5.1 and 5.2 we present the results for Eventual Collision Freedom, and in Section 5.3 we discuss the results for systems with unrestricted collisions.

5. CONSENSUS ALGORITHMS

In the consensus problem, each node in P starts with an input value from a fixed set V, and outputs a decision value so that the following is satisfied: (1) *Agreement*: No two correct nodes in P decide on different values; (2) *(Strong) Validity*: If a node in P decides a value $v \in V$, then v is the initial value of a node in P; and (3) *Termination*: All correct nodes in P eventually decide. In Section 7, we will consider the following weaker validity property: *Weak Validity*: If $v \in V$ is an input value of some node in P, then there exists an execution where v is decided.

In this section, we show how to solve consensus using eventually accurate collision detectors and a wake-up service. Our results are summarized in Table 2.

To simplify the presentation, in the following we will use the term *Earliest Stabilization Time (EST)* to refer to round $r = \min\{r' \geq \max\{r_{ecf}, r_{wake}, r_{acc}\}\}$.

5.1 Consensus: Reliable Collision Detectors

The pseudo-code in Algorithm 1 is an implementation of consensus using a collision detector in $\diamond\mathcal{AC}$ (and by extension $\mathcal{AC}$). The algorithm tolerates any number of node failures and terminates in at most five rounds after EST.

The algorithm consists of two phases: a *proposal* phase and a *veto* phase. In the proposal phase, every active node sends out its estimate. The passive nodes do not broadcast. If a node hears no collisions, it updates its estimate to the minimum value received. If a node detects a collision, or if a node hears more than one estimate, then it performs a veto in the second phase. If in the veto phase there are no veto messages received or collisions detected, then a node can decide.

Algorithm 1: An adaptive consensus algorithm with a $\diamond\mathcal{AC}$ collision detector.

```
1  Process P_i:
2    estimate ← the initial value of process P_i
3    phase ← proposal
4    For each round r, r ≥ 1 do:
5      if (phase = proposal) then
6        Let active be the advice of the wake-up service
7        if active then bcast(estimate)
8        messages ← recv()
9        if (± ∉ messages) then
10         estimate ← min{v ∈ messages}
11       phase ← veto
12     else if (phase = veto) then
13       if (± ∈ messages) or (|messages| > 1) then
14         bcast(veto)
15       veto-messages ← recv()
16       if (veto-messages = ∅) then
17         if (|messages| = 1) then
18           decide(estimate) and halt
19       phase ← proposal
```

THEOREM 1. *Algorithm 1 is an implementation of consensus for nodes augmented with a collision detector in $\diamond\mathcal{AC}$. It terminates in at most 5 rounds after EST.*

PROOF (SKETCH). Let r be the earliest round in which a node decides, and let p_i be a node that decides in round r. In proposal round $r-1$, node p_i receives a message from every active node, since it receives no collision notifications. Moreover, every message contains the same estimate. Since p_i heard no messages or collisions in the veto round r, every other non-failed node must also have received every message in round $r-1$ and updated its estimate. Therefore p_i's decision value is the only possible decision value.

Next, we show termination. Eventual collision freedom, eventual accuracy, and Property 2 imply that eventually the system reaches EST, at which point there are fewer than b active nodes. During these rounds, the first proposal phase results in every participant choosing an estimate, and after the second proposal phase no node vetoes, hence every node decides. □

5.2 Consensus: Unreliable Collision Detectors

In this section we consider consensus protocols for nodes augmented with an unreliable collision detector.

Consensus with maj-$\diamond\mathcal{AC}$ collision detectors

We show that Algorithm 1 is correct with a collision detector in maj-$\diamond\mathcal{AC}$:

THEOREM 2. *Algorithm 1 is an implementation of consensus for nodes augmented with a collision detector in maj-$\diamond\mathcal{AC}$. It terminates in at most 5 rounds after EST.*

PROOF (SKETCH). As in the case of a $\diamond\mathcal{AC}$ collision detector, if any node broadcasts during the veto phase, no node will decide in that round, since every node either receives a majority of the messages broadcast (which are all veto messages), or a collision notification.

As in Theorem 1, consider the first round, r, at which any node decides. Since no node performs a broadcast during the veto phase r, every node receives only a single estimate — and no collision notifications — during the proposal phase $r-1$. Moreover, each node receives a majority of the messages broadcast in round $r-1$, since maj-$\diamond\mathcal{AC}$ detects when $\geq$ half the messages are lost. Since every majority set intersects, all received the same unique estimate. Therefore, at the end of round $r-1$ all participants adopt the same estimate. It is therefore easy to see that no other decision value is possible.

Termination follows as in the case of $\diamond\mathcal{AC}$, since once the wake-up service provides good advice, collisions cease during the proposal rounds. □

Algorithm 2: An adaptive consensus algorithm with a 0-$\diamond\mathcal{AC}$ collision detector.

```
1  Process P_i:
2    estimate ← the initial value of process P_i
3    phase ← prepare
4    size ← number of bits used to represent initial values
5    For each round r, r ≥ 1 do:
6      if (phase = prepare) then
7        if (active) then bcast(estimate)
8        messages ← recv()
9        if (|messages - {±}| > 0) then
10         estimate ← min{v ∈ messages}
11       decide ← true
12       bit ← 1
13       phase ← propose
14     else if (phase = propose) then
15       if (not decide) or (estimate[bit] = 1) then
16         bcast(veto)
17       messages ← recv()
18       if (|messages| > 0) then
19         if (estimate[bit] = 0) then
20           decide ← false
21       bit ← bit + 1
22       if (bit > size) then phase ← accept
23     else if (phase = accept) then
24       if (not decide) then bcast(veto)
25       messages ← recv()
26       if (decide = true) and (|messages| = 0) then
27         decide(estimate) and halt
28       phase ← prepare
```

Consensus with 0-$\diamond\mathcal{AC}$ collision detectors

We now present Algorithm 2 which solves consensus with a collision detector in 0-$\diamond\mathcal{AC}$. It terminates in $O(log|V|)$ rounds after EST. In Section 6, we show that this lower bound is tight.

Algorithm 2 has three phases. In the first phase, each active node proposes an estimate. Every node adopts the minimum estimate it receives, resolving to reject if it hears any collisions or more than one value. In the second phase, the nodes attempt to check that they all have the same estimate. There is one round for each bit in the estimate; if a node has an estimate with a one in the bit associated with that round, then it broadcasts a message. If a node has an estimate with a zero in the bit associated with that round, it listens for broadcasts, and decides to reject if it hears any broadcasts or collisions. Finally, the nodes enter the accept phase. In this phase, any node that wants to

reject broadcasts a veto. If any node performs a veto, then all the nodes return to the propose phase and start again.

THEOREM 3. *Algorithm 2 solves consensus for nodes augmented with* 0-$\diamond\mathcal{AC}$ *and terminates* $2(\log|V|+2)$ *rounds after EST.*

PROOF (SKETCH). If node p_i decides v in round r, then all nodes have estimate value v at the end of round r. All nodes must have began round r with $decide = true$, or else they would have broadcast a veto and p_i would have received at least one message or a collision notification, leading p_i not to decide. If all nodes began round r with $decide = true$, then all nodes broadcast on the same schedule during the preceding propose rounds, therefore all nodes most have the same estimate value v. Termination is straightforward, as soon as eventual collision freedom, eventual accuracy, and good advice hold. □

5.3 Collision-resistant consensus protocols

It is a natural question to ask whether some collision detector classes can be powerful enough to solve consensus even in the face of unrestricted message loss. Surprisingly, the answer to this question is yes. A simple variant of Algorithm 2 can be used to solve strong validity consensus in $O(log|V|)$ rounds with a collision detector in $\mathcal{AC}$.

In particular, unrestricted message loss poses a problem only for the *prepare* phase of Algorithm 2. If we cannot guarantee a collision-free *prepare* round, we cannot guarantee liveness. To circumvent this issue, we replace this existing phase with code that performs a binary search through the domain of all possible initial values. At each iteration of the search, we allot one round for each of the two subsets that we can possibly recurse on. Nodes only broadcast in rounds corresponding to subsets that contain their initial value. If noise (message or collision notification) is heard for both subsets in a given iteration, then the algorithm always chooses to recurse on the first subset. If no noise is heard for either subset of a given split (e.g. as the result of node failures), the search starts from scratch by returning to the full set of values in the next iteration.

6. LOWER BOUNDS

In this section, we show lower bounds that match the upper bounds of the previous section. We first examine a collision detector called *half-complete-$\mathcal{AC}$* that is always accurate and guarantees to deliver a collision only if the number of messages received in a round r is strictly less than $M(r)/2$, where $M(r)$ is the number of messages brodcast in r. We show that with a half-complete-$\mathcal{AC}$ collision detector, consensus cannot be solved in a constant number of rounds. This demonstrates that only a slight weakening of maj-completeness results in a substantial complexity gap. It also implies that Algorithm 2 is optimal.

We then consider the case where collisions never abate. In this case, we show that it is impossible to solve consensus without (permanent) accuracy, and then show that even with (permanent) accuracy consensus cannot be solved in a constant number of rounds. Together, these results show that the algorithm described in Section 5.3 is optimal.

Tightness of bounds in Section 5.2

We show that, no algorithm (where the nodes do not have unique ids) can solve consensus in a constant number of rounds after EST if half or more of the messages sent in a round can be lost without detection. To obtain the strongest possible lower bound, we assume that the nodes have access to a wake-up service (see Section 3), and to a collision detector, called *half-complete-$\mathcal{AC}$*, which is always accurate and guarantees to deliver a collision only if the number of messages received in a round r is strictly less than $M(r)/2$, where $M(r)$ is the number of messages brodcast in r. We prove the following

THEOREM 4. *Let A be an algorithm that solves consensus with a wake-up service satisfying Property 2 and a collision detector in half-complete-$\mathcal{AC}$. Assume w.l.o.g. that $|V| > 2$. Then, there exists an execution of A where $EST = 1$ and the nodes do not decide before round* $\log(|V|)$.

We first introduce some definitions: Given a k-round execution α, we define the *transmission schedule* of node p_i in α, denoted $ts(\alpha, i)$, to be the sequence of 0s and 1s of length k, such that the jth element of $ts(\alpha, i)$ is 1 iff p_i transmits a message in round j. If all the nodes in α follow the same transmission schedule, then we refer to this common schedule as $ts(\alpha)$. We say that two executions α and β are equivalent w.r.t. to their transmission schedules, denoted $\alpha \equiv \beta$, if all the nodes in α and β follow the same transmission schedule, and $ts(\alpha) = ts(\beta)$. The result follows from the following key lemma:

LEMMA 4.1. *For each k, $1 \leq k \leq \log(|V|) - 1$, let $\mathcal{A}_k$ denote the set of all the k-round executions of A. Let Π_k be the partition of $\mathcal{A}_k$ to the equivalence classes w.r.t. the relation $\equiv$. Then, $\Pi_k \neq \emptyset$, and each $P \in \Pi_k$ contains at least two executions α and β satisfying the following:*

1. *Both α and β consist of disjoint sets of nodes, denoted L and R respectively, such that $|L| = |R|$.*
2. *All the nodes in L (resp. R) start with the same initial value v (resp. w), and $v \neq w$.*
3. *No messages are lost, no collisions are detected, all the nodes are correct and the wake-up service outputs are the same at all nodes in both α and β.*
4. *There exists a k-round execution γ consisting of exactly the nodes in $L \cup R$ such that the nodes in L (resp. R) receive the same set of messages as that received in α (resp. β), and no collision notifications.*
5. *No node decides in α, β and γ.*

PROOF (SKETCH). The proof is by induction on k. For $k = 1$, consider the set P_v of all the 1-round executions where the nodes are correct and start with the same initial value v, no messages are lost, no collisions are detected and the wake-up service outputs are the same at all nodes in every round. Since all the nodes have the same initial state in $\alpha_v \in P_v$, they all will take a consistent decision as to whether to transmit a message or not. Moreover, since $|V| > 2$, there exist a value $w \in V$, $w \neq v$, and a pair of executions $\alpha_v \in P_v$ and $\alpha_w \in P_w$ such that the sets of nodes L and R participating in α_v and α_w are of equal size and disjoint, and $ts(\alpha_v) = ts(\alpha_w)$. We then construct γ as required by the lemma statement. Since the collision detector satisfies half-completeness, the nodes in L (resp. R) can loose the

messages sent by the nodes in R (resp, L). Hence, γ is a valid execution of A. Finally, no node can decide in either α, β or γ, as otherwise, the nodes in L must decide v, and the nodes in R must decide $v \neq w$ violating agreement.

For the inductive step $k > 1$, we notice that as long as $k < \log(|V|)$, it is always possible to find two executions α_v and α_w, $v \neq w$, with the same transmission schedules belonging to some equivalence class in Π_{k-1} that can be extended by one round. Indeed, for $k < \log(|V|)$, there are at most $|V|/2$ transmission schedules to follow for the first k rounds. Since there are $|V|$ initial values, and all the executions where the nodes start with the same initial value follow the same transmission schedule, there must be two executions α_v and α_w, $v \neq w$, that follow the same transmission schedule. The rest of the proof is similar to the base case proof. □

PROOF (THEOREM 4). The execution α constructed in Lemma 4.1 is indistinguishable to the nodes in L from an execution α' which is identical to α except $b = |L|$. In turn, α' is identical to some execution where $EST = 1$. The result follows. □

Tightness of bounds in Section 5.3

In this section, we show that it is impossible to solve consensus without eventual collision freedom if a collision detector does not satisfy (perpetual) accuracy.

THEOREM 5. *There does not exist an algorithm that solves 1-resilient consensus with collision detector in $\diamond\mathcal{AC}$ and a wake-up service if the communication layer does not guarantee collision freedom (i.e., the message loss is completely unrestricted) and the set of participants is a priori unknown.*

PROOF (SKETCH). Assume by contradiction that such algorithm A exists. Let S be the set of nodes participating in A and assume that at least two nodes in S are correct. We construct an execution α of A as follows: Partition the nodes in S into two sets S_1 and S_2 each of which including at least one correct node, and the nodes in S_1 (resp. S_2) starting with v_1 (resp. v_2) where $v_1 \neq v_2$. In every round of α, let each node in S_1 (resp. S_2) to loose all the messages sent by the nodes in (and only in) S_2 (resp. S_1), and to detect a collision. We claim that no node can decide in α. Indeed, for each k-round prefix α_k of α, there exists an execution $\beta_{1,k}$ (resp. $\beta_{2,k}$) where all the nodes in S_2 (resp. S_1) are crashed from the beginning; in the first k rounds of $\beta_{1,k}$ (resp. $\beta_{2,k}$) all the nodes in S_1 (resp. S_2) receive exactly the same set of messages and collision notifications as in α_k; and the $EST = k + 1$. (Note that both $\beta_{1,k}$ and $\beta_{2,k}$ are valid executions of A since the collision detector is allowed to be inaccurate before EST.) Then, no node in S can decide after α_k since otherwise, all the nodes in S_1 (resp. S_2) will decide the same value as the one decided in $\beta_{1,k}$ (resp. $\beta_{2,k}$) violating agreement. □

Finally, we can use a similar argument as that used to prove Theorem 4, to show that the following result holds (the proof can be found in the full version):

THEOREM 6. *Let A be an algorithm that solves consensus with a collision detector in $\mathcal{AC}$, and suppose that the communication layer does not guarantee collision freedom. Assume w.l.o.g. that $|V| > 2$. Then, there exists an execution of A where the nodes do not decide before round* $\log(|V|)$.

7. WEAK-VALIDITY CONSENSUS

If consensus is only required to satisfy weak validity, then it is possible to overcome some of the lower bounds discussed in Section 6. In particular, in this section, we describe two algorithms that do not require collision freedom (Property 1). The first algorithm uses a collision detector in $\mathcal{AC}$ and terminates in constant rounds, and the second one uses a collision detector in 0-$\mathcal{AC}$ and terminates in $O(\log |V|)$ rounds.

Recall that weak validity only requires that there exists an execution in which the decision is an initial value of some participant. In particular, node's may decide on a default value (even though that value may not be any node's initial value). Consider, for example, a transactional database where the default decision may be to abort the transaction. In a collision-free execution, the initial value of some node will be chosen; otherwise, the default value may be chosen.

A minor variant of Algorithm 1 solves weak-validity consensus in two rounds with a collision detector in $\mathcal{AC}$. Each node executes the proposal and veto phases, as previously described in Section 5.1. Recall that in Algorithm 1 if a node detects a veto, then it repeats the two phases of the protocol. For the weak-validity consensus, however, there is no need to repeat the protocol; instead, if a node receives a veto, then it simply decides on the default value. With a collision detector in $\mathcal{AC}$, this ensures agreement: a node only chooses the default value when it detects a veto; this implies that some node detected a collision in the proposal phase and broadcast a veto; therefore every participant must detect a veto and choose the default value.

Similarly, a minor variant of Algorithm 2 solves weak-validity consensus using collision detectors in 0-$\mathcal{AC}$. It requires $O(\log |V|$ rounds to complete, where V is the set of possible initial values. Again, for weak validity, if a node detects a veto in the accept phase, then it simply decides on the default value, instead of repeating the protocol.

8. PERFORMANCE EVALUATION

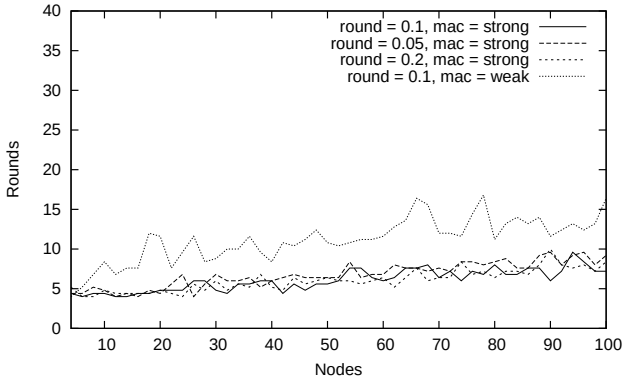
In this section we evaluate the performance of our algorithms by simulation. First, we examine Algorithm 1 under different MAC layer conditions. Second, we examine a multi-hop consensus protocol based on Algorithm 1, and then compare it to a simple flood-and-gossip solution.

In our expertiments, we used the ns-2 network simulator [15][1] with integrated CMU wireless extensions [30]. We modified the CMU 802.11 MAC layer implementation to generate collision notifications for incoming messages lost due to interference. Note that we used our MAC layer only in broadcast mode, which, unlike 802.11 unicast communication, does not employ RTS/CTS handshaking. In the single-hop scenarios, our collision detector behaved as $\mathcal{AC}$. In the multi-hop case, due to colliding messages originating from nearby regions, the collision detector behaved as $\diamond\mathcal{AC}$. The transmission range of each node was approximately 20 meters, and the two-ray ground reflection model was used to achieve realistic radio propagation effects.

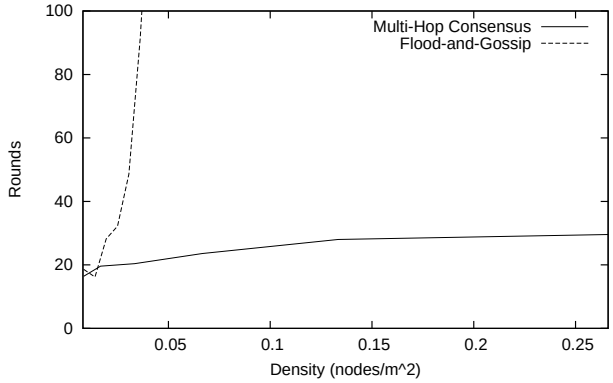
8.1 The Wake-up Service

For the purposes of simulation, we implement a wake-up service using a simple approximation of a well-known back-off strategy [17,20,31,41]. For each round r during which the

[1]Release version 2.27



(a) Average number of rounds needed to reach consensus for Algorithm 1 under varying densities and MAC layer tolerances.



(b) Average number of rounds needed to reach multi-hop consensus in a 5-hop network with increasing node density.

Figure 1: Simulation results with ns-2 using 802.11 wireless MAC layer augmented with collision detection. Each data point is the average of five independent simulation runs.

wake-up service is queried, (1) if p_i's wake-up service detects a collision in r, then with probability 1/2, it recommends p_i to become passive the next time the service is queried. (2) if p_i does not detect any broadcast activity in r, then with probability 1/2, it recommends p_i to become active the next time the service is queried. (Some slight modifications would be needed for unreliable collision detectors.) Using a straightforward Chernoff bound, it is easy to show that if there are n nodes in the execution, the wake-up service achieves EST within $O(\log^2 n)$ rounds after $\max(r_{ecf}, r_{acc})$, with high probability.

8.2 Single-hop Consensus

Figure 1.a plots the number of rounds required to reach consensus for Algorithm 1, described in Section 5.1. Even as the density of the deployment increases, the number of rounds to decide remains almost constant. In order to test adaptivity to different MAC layers (and ensure that our simulated MAC layer was not simply a special case), we varied the MAC layer parameters, running simulations with three different round lengths. We also tested our protocol on top of a "weak" collision avoidance scheme in which the back-off/carrier sensing features of 802.11 were disabled, leaving only a simple initial randomized broadcast delay. This was designed to represent the minimal MAC layer that might be used by real devices. These changes had little effect on the algorithm performance.

8.3 Multi-hop Consensus

To demonstrate the utility of Algorithm 1 in challenging environments with lots of noise and numerous unrelated, interfering broadcasts, we used it to implement a multi-hop consensus protocol. The multi-hop scenario rigorously tests the collision-tolerance properties of the single-hop algorithm: since all the nodes are running the same single-hop algorithm, the interference is exactly synchronized.

Our solution for multi-hop consensus proceeds as follows. The network is divided into a series of non-overlapping grid squares. Every node knows the pattern of grid squares and its approximate location in the grid. In practice, this is a reasonable assumption, as the localization problem in wireless ad hoc networks is well studied [28, 32, 33, 40]. All nodes within a given grid square are within communication range of each other. First, single-hop consensus is conducted for each grid square using Algorithm 1. Second, all nodes execute a *Grid Consensus* algorithm that gossips the grid square consensus values throughout the network – using the wake-up service to reduce contention. Once a node has received a value for every grid square, it can decide by applying a deterministic function to this set. (Please see the full version [11] of this paper for the Grid Consensus pseudo-code, and a more detailed presentation of the multi-hop model and the correctness proofs.)

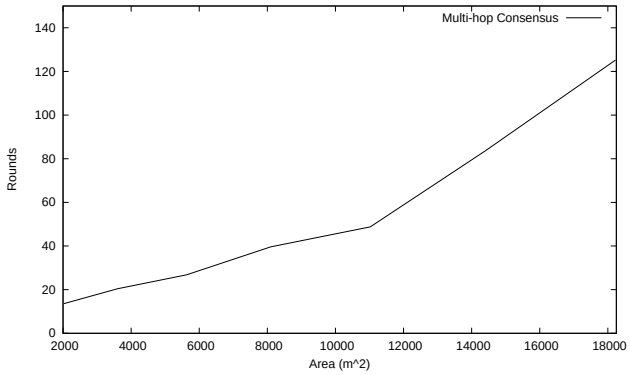


Figure 2: Average number of rounds needed to reach multi-hop consensus for a density of 0.02667 $nodes/m^2$ (approx. 6 nodes per single-hop area) and increasing network area.

We compared our algorithm against a simple flood-and-gossip strategy similar to [2, 25, 26]. Nodes decided to flood their initial value with probability 0.2, and the algorithm was considered terminated once all the nodes had received every value that had been broadcast.

We evaluated our solutions in a $3600m^2$, 5-hop diameter network divided into sixteen non-overlapping single-hop grid squares. Figure 1.b shows the number of rounds required for multi-hop consensus in this environment under increasing

node density. The stability of our solution is notable: As the density increases from an average of 2 nodes per single-hop grid square (0.00125 $nodes/m^2$), to an average of 63 nodes per grid square (0.27778 $nodes/m^2$), multi-hop consensus consistently terminates in 15 to 30 rounds even when as many as 960 nodes are participating. In contrast, the flood-and-gossip approach worked reasonably well for small densities, but at larger values it was overwhelmed by the volume of messages traveling throughout the network.

We also studied the performance of the multi-hop protocol in larger networks. For a fixed density of 0.02667 $nodes/m^2$ (approximately 6 nodes per single-hop area), we varied the network diameter from 4-hops to 10-hops, with the largest network tested featuring 1000 nodes scattered over an area roughly the size of three American football fields placed side-by-side. The results, as described in Figure 2, show that up to an area of 11000 m^2 the number of rounds needed to decide increase at a reasonable rate of one round for every 300 m^2 of area added. After this point, the rate increases to one additional round for every 100 m^2 of area added.

A careful analysis attributes this rate increase to a failure of our wake-up service implementation in larger networks. Specifically, we deployed our nodes randomly to achieve the fixed average density. Accordingly, the larger networks were more likely to contain a few single-hop grid squares, often near the borders, that contained a very small number of nodes (i.e., < 3).

Our wake-up service implementation, for both the local and multi-hop phase of our solution, does not handle these low density regions efficiently. We found that by increasing the aggressiveness of our service in these border squares (for example, increase the probability that a node in a border square considers itself active) we could gain better performance for large networks. Such changes, however, degrade performance for smaller networks. The best solution seems to be an adaptive wake-up service that behaves differently depending on the network dimensions and the node's known location in the overlay grid. We leave the investigation of such an adaptive service for future work.

9. CONCLUDING REMARKS

In this paper we investigated the solvability of consensus in wireless ad hoc networks under a realistic collision-prone model with an unknown number of participants. We presented solutions with efficiency varying with the quality of collision detection available, and we showed that our bounds are tight. We believe that our results will impel the physical layer radio designers to appreciate the importance of exporting collision detection information to higher levels of protocol stack.

We considered crash failure of nodes only. In the future, we intend to investigate consensus in the presence of Byzantine nodes. Moreover, we hope to investigate other algorithms in both single-hop and multi-hop collision-aware models, and we will corroborate the efficiency of our algorithms by experimenting with real wireless sensor network [37] deployments.

Acknowledgments

We would like to thank Nancy Lynch for conversations that inspired many of the results in this paper, and Daniela Tulone for discussions about randomized wake-up services.

10. REFERENCES

[1] IEEE 802.11. Wireless LAN MAC and physical layer specifications, June 1999.

[2] K. Akkaya and M. Younis. A survey of routing protocols in wireless sensor networks. *Elsevier Ad Hoc Network Journal*, 3(3):325–349, 2005.

[3] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. On the complexity of radio communication. In *STOC: Symposium on Theory of Computing*, pages 274–285. ACM Press, 1989.

[4] J. Aspnes, F. Fich, and E. Ruppert. Relationships between broadcast and shared memory in reliable anonymous distributed systems. In *18th International Symposium on Distributed Computing*, pages 260–274, 2004.

[5] R. Bar-Yehuda, O. Goldreich, and A. Itai. Efficient emulation of single-hop radio network with collision detection on multi-hop radio network with no collision detection. *Distributed Computing*, 5:67–71, 1991.

[6] R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization. *Journal of Computer and System Sciences*, 45(1):104–126, 1992.

[7] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, 1996.

[8] B. Chlebus and D. Kowalski. A better wake-up in radio networks. *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 266–274, 2004.

[9] B. S. Chlebus, L. Gasieniec, A. Gibbons, A. Pelc, and W. Rytter. Deterministic broadcasting in ad hoc radio networks. *Distributed Computing*, 15(1):27–38, 2002.

[10] G. Chockler, M. Demirbas, S. Gilbert, N. Lynch, C. Newport, and T. Nolte. Reconciling the theory and practice of unreliable wireless broadcast. *International Workshop on Assurance in Distributed Systems and Networks (ADSN)*, 2005. To appear.

[11] G. Chockler, M. Demirbas, S. Gilbert, C. Newport, and T. Nolte. Consensus and collision detectors in wireless ad hoc networks. Technical Report 980, MIT CSAIL, 2005.

[12] A. Clementi, A. Monti, and R. Silvestri. Round robin is optimal for fault-tolerant broadcasting on wireless networks. *J. Parallel Distributed Computing*, 64(1):89–96, 2004.

[13] J. Deng, P. K. Varshney, and Z. J. Haas. A new backoff algorithm for the IEEE 802.11 distributed coordination function. In *Communication Networks and Distributed Systems Modeling and Simulation (CNDS '04)*, 2004.

[14] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM*, 35(2):288–323, 1988.

[15] K. Fall and K. Varadhan. *The ns Manual*, April 2002. www.isi.edu/nsnam/ns/ns-documentation.html.

[16] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985.

[17] R. Gallager. A perspective on multiaccess channels. *IEEE Trans. Information Theory*, IT-31:124–142, 1985.

[18] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex behavior at scale: An experimental study of low-power wireless sensor networks. *UCLA Computer Science Technical Report UCLA/CSD-TR*, 2003.

[19] J. F. Hayes. An adaptive technique for local distribution. *IEEE Trans. Commun.*, 26(8):1178–1186, 1978.

[20] S. Olariu K. Nakano. A survey on leader election protocols for radio networks. *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN)*, pages 71–79, 2002.

[21] C-Y. Koo. Broadcast in radio networks tolerating byzantine adversarial behavior. *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 275–282, 2004.

[22] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott. Experimental evaluation of wireless simulation assumptions. In *Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 78–82, 2004.

[23] E. Kranakis, D. Krizanc, and A. Pelc. Fault-tolerant broadcasting in radio networks. In *Proceedings of the 6th Annual European Symposium on Algorithms*, pages 283–294, 1998.

[24] L. Lamport. Paxos made simple. *ACM SIGACT News*, 32(4):18–25, 2001.

[25] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. *First USENIX/ACM Symposium on Networked Systems Design and Implementation*, 2004.

[26] C. Livadas and N. Lynch. A reliable broadcast scheme for sensor networks. Technical Report MIT-LCS-TR-915, MIT CSAIL, 2003.

[27] N. Lynch. *Distributed Algorithms*. Morgan Kaufman, 1996.

[28] K. Mechitov, S. Sundresh, Y-M. Kwon, and G. Agha. Cooperative tracking with binary-detection sensor networks. Technical Report UIUCDCS-R-2003-2379, University of Illinois at Urbana-Champaign, 2003.

[29] R. M. Metcalfe and D. R. Boggs. Ethernet: distributed packet switching for local computer networks. *Commun. ACM*, 19(7):395–404, 1976.

[30] CMU Monarch. The CMU Monarch Project's Wireless and Mobility Extensions to NS, 1998.

[31] K. Nakano and S. Olariu. Uniform leader election protocols in radio networks. In *ICPP '02: Proceedings of the 2001 International Conference on Parallel Processing*, pages 240–250. IEEE Computer Society, 2001.

[32] D. Niculescu and B. Nath. Ad hoc positioning system (APS) using AOA. *IEEE INFOCOM The Conference on Computer Communications*, 22(1):1734–1743, 2003.

[33] D. Niculescu and B. Nath. DV based positioning in ad hoc networks. *Kluwer journal of Telecommunication Systems*, 22(1–4):267–280, 2003.

[34] J. Polastre and D. Culler. Versatile low power media access for wireless sensor networks. *The Second ACM Conference on Embedded Networked Sensor Systems (SENSYS)*, pages 95–107, 2004.

[35] N. Santoro and P. Widmayer. Time is not a healer. In *Proceedings of the 6th Annual Symposium on Theoretical Aspects of Computer Science*, pages 304–313. Springer-Verlag, 1989.

[36] N. Santoro and P. Widmayer. Distributed function evaluation in presence of transmission faults. *Proc. Int. Symp. on Algorithms (SIGAL)*, pages 358–367, 1990.

[37] Crossbow Technology. Mica2. `www.xbow.com/Products/Wireless_Sensor_Networks.htm`.

[38] B. S. Tsybakov and V. A. Mikhailov. Free synchronous packet access in a broadcast channel with feedback. *Prob. Inf. Transmission*, 14(4):1178–1186, April 1978.

[39] T. van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. *The First ACM Conference on Embedded Networked Sensor Systems (SENSYS)*, pages 171–180, 2003.

[40] K. Whitehouse. The design of Calamari: an ad-hoc localization system for sensor networks. Master's thesis, U.C. Berkeley, 2002.

[41] D. E. Willard. Log-logarithmic selection resolution protocols in a multiple access channel. *SIAM Journal of Computing*, 15(2):468–477, 1986.

[42] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of multihop routing in sensor networks. *The First ACM Conference on Embedded Networked Sensor Systems (SENSYS)*, pages 14–27, 2003.

[43] A. Woo, K. Whitehouse, F. Jiang, J. Polastre, and D. Culler. Exploiting the capture effect for collision detection and recovery. *The Second IEEE Workshop on Embedded Networked Sensors (EmNetS-II)*, May 2005.

[44] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2002.

[45] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. *The First ACM Conference on Embedded Networked Sensor Systems (SENSYS)*, pages 1–13, 2003.