Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2006-026 April 19, 2006

Of Malicious Motes and Suspicious Sensors

Seth Gilbert, Rachid Guerraoui, and Calvin Newport

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Of Malicious Motes and Suspicious Sensors

Seth Gilbert
MIT CSAIL
sethg@mit.edu

Rachid Guerraoui
EPFL IC and MIT CSAIL
rachid.guerraoui@epfl.ch

Calvin Newport
MIT CSAIL
cnewport@mit.edu

Abstract

How much damage can a malicious tiny device cause in a single-hop wireless network? Imagine two players, Alice and Bob, who want to exchange information. Collin, a malicious adversary, wants to prevent them from communicating. By broadcasting at the same time as Alice or Bob, Collin can destroy their messages or overwhelm them with his own malicious data. Being a tiny device, however, Collin can only broadcast up to β times. Given that Alice and Bob do not know β, and cannot distinguish honest from malicious messages, how long can Collin prevent them from communicating? We show the answer to be $2\beta + \Theta(\lg |V|)$ communication rounds, where V is the set of values that Alice and Bob may transmit. We prove this result to be optimal by deriving an algorithm that matches our lower bound—even in the stronger case where Alice and Bob do not start the game at the same time.

We then argue that this specific 3-player game captures the general extent to which a malicious adversary can disrupt coordination in a single-hop wireless network. We support this claim by deriving—via reduction from the 3-player game—round complexity lower bounds bounds for several classical n-player problems: $2\beta + \Theta(\lg |V|)$ for reliable broadcast, $2\beta + \Omega(\log n/k)$ for leader election among k contenders, and $2\beta+\Omega(k \lg |V|/k)$ for static k-selection. We then consider an extension of our adversary model that also includes up to t crash failures. We study binary consensus as the archetypal problem for this environment and show a bound of $2\beta + \Theta(t)$ rounds. We conclude by providing tight, or nearly tight, upper bounds for all four problems. The new upper and lower bounds in this paper represent the first such results for a wireless network in which the adversary has the ability to disrupt communication.

1 Introduction

Ad hoc networks of wireless devices hold significant promise for the future of ubiquitous computing. Such networks are appealing because they are cost-effective and easy to deploy in a wide variety of contexts. At the same time, however, they are particularly vulnerable to adversarial interference due to their use of a shared, public communication medium and their typical deployment in unprotected environments. In this paper, we ask the fundamental question of how much disruption a malicious (Byzantine) device can cause in such a network. We first establish a tight bound for a constrained 3-player game, and then show how this bound implies results for more general n-player problems such as reliable broadcast, leader election, static k-selection and consensus.

The Environment. We consider a single-hop wireless radio channel shared by several devices operating in synchronous rounds. If one device broadcasts, then all devices receive its message. If two or more devices broadcast, then each listening device either receives one of the messages or simply detects noise, indicating a collision. The honest devices, which we call *players*, follow a designated algorithm, and can use various strategies—such as a round-by-round TDMA schedule—to avoid contention on the communication channel.

A major novelty of our work is considering, for the first time in the context of wireless networks, malicious devices that can deviate from honest broadcasting behavior. In our model, a malicious device can broadcast in any round, potentially destroying honest messages or overwhelming them with malicious data. As is typically assumed in these networks—due to computational and bandwidth limitations, as well as the challenges of ad hoc deployment—the tiny devices cannot authenticate messages. Therefore, a message sent by a malicious device cannot necessarily be distinguished from one sent by an honest player.

The Game. We first focus on a constrained communication game involving two honest players—Alice and Bob—and one malicious adversary, Collin (the *Collider*). Alice and Bob are each initially given a value to communicate. Collin, jealous about the state of affairs between Alice and Bob, is determined to prevent them from communicating (in either direction) for as long as possible. A straightforward strategy for Collin is to broadcast a message in every round, systematically provoking a collision whenever another message is sent, thus preventing Alice and Bob from ever exchanging any information.

Being a tiny device, Collin cannot broadcast more than some budget of β messages, where β is *a priori* unknown to Alice and Bob. This assumption generalizes several examples of limited malicious interference, such as a bounded energy budget.

The Tight Bound. Collin's strategy of provoking systematic collisions can obviously delay Alice and Bob from communicating for at least β rounds. This paper shows, however, that Collin can do better than this simple "jamming" approach. With a budget of β messages, Collin can actually delay Alice and Bob's communication for at least $2\beta + \Omega(\lg |V|)$ communication rounds, where V is the set of possible values that Alice and Bob may communicate. More precisely, we show the following:

1. *Lower bound.* Collin can prevent *both* (a) Alice from outputting Bob's initial value *and* (b) Bob from outputting Alice's initial value, for $2\beta + \lg |V|/2 - 1$ rounds. We prove the lower bound (in Section 4) by exhibiting a strategy for Collin to delay Alice and Bob, exploiting the fact that they can never trust

any message, since Collin could *overwhelm* it with a fake message of his. As a consequence, the only rounds in which Collin needs to send a message are the rounds that are otherwise *silent*, i.e., those in which Alice and Bob choose to encode information by *not* sending a message.

2. *Upper bound.* There exists a protocol that allows Bob to output Alice's initial value after $2\beta + O(\lg |V|)$ rounds. This bound holds even if Alice and Bob begin the game in different rounds. We describe our protocol (in Section 5) as a combination of a basic sub-protocol in which Alice repeatedly transmits a single bit of information to Bob using alternating *data* rounds and *veto* rounds, together with a *synchronization* sub-protocol that compensates for the differing start times of Alice and Bob.

As mentioned, key to our *lower bound* is the fact that Alice and Bob cannot necessarily distinguish a message from Collin from an honest message. If they could, the tight bound is clearly β rounds. Thus, one interpretation of our result is that authentication, which can be expensive in wireless ad hoc networks, should be deployed only if its cost is less than the cost of delaying communication by β additional rounds, where β represents the message budget of an adversary. On the other hand, key to our *upper bound* protocol is the ability for Alice and Bob to distinguish a "silent" round from a round in which a collision occurred, e.g., Alice and Bob can detect some electromagnetic noise in the latter case. Without this capability, a silent round cannot be used to encode information and Collin can prevent all communication forever. In this sense, the absence of collision detection provides Collin with infinite power.

Another interesting interpretation of our tight bound can be given in terms of the cost of sending a message as compared to listening. Assume that it takes s units of energy to send a message, and ℓ units of energy to listen for a message.[1] If Alice, Bob, and Collin all begin with the same amount of energy, then our result implies the following corollary: *Alice and Bob can communicate if and only if* $s > 2\ell$. In other words, communication can be made more robust against malicious devices if the cost of broadcasting is high. In practice, one might also distinguish the cost of sending a message, K, from the cost of causing a collision, k, where k would typically be smaller than K. In this sense, our tight bound indicates that, if $k < K/2$, then the best strategy for Collin is to jam Alice and Bob with collisions; if $k \geq K/2$, then the best strategy for Collin, on the contrary, is to follow the silence-filling strategy underlying our lower bound.

[1] In real systems, the ratio of s to ℓ varies depending on the network configuration and the underlying hardware. For example, by filtering signals below a certain energy threshold, a network designer can increase the cost of sending a message.

n-Player Implications. We argue that the trials and tribulations of Alice and Bob capture something fundamental about the ability of malicious devices to disrupt coordination in a wireless network. We support this claim by deriving (in Section 6) new lower bounds—via reduction from our 3-player game—for several classical n-player problems: reliable broadcast, leader election, static k-selection and consensus. These represent, to the best of our knowledge, the first bounds for these problems in a wireless network with an adversary that can arbitrarily disrupt communication.

We first consider a strict generalization of our 3-player framework, featuring n players and a malicious adversary whose power is reflected in his budget of β messages. It is without loss of generality that we assume *one* malicious device: if there are several colluding devices, β represents the total number of broadcasts that they can collectively use. We then describe an algorithm that can be used by Alice and Bob to simulate n honest players executing an arbitrary protocol, and use this simulation to derive the following (round complexity) lower bounds: reliable broadcast: $2\beta+\Theta(\lg |V|)$; leader election[2]: $2\beta+\Omega(\log \frac{n}{k})$; static k-selection: $2\beta + \Omega(k \lg \frac{|V|}{k})$. For the latter two cases, k represents the number of participants contending, either to become leader or to transmit their initial value.

We next consider a more general framework with n players and a malicious adversary who can not only send β messages, but can also *crash* up to t honest devices. We study binary consensus as an archetypal problem in this framework, and derive a lower bound of $2\beta + \Theta(t)$ rounds. This result might be surprising because, at first glance, it is not obvious that crashed processes can delay a decision in a synchronous broadcast medium. (For example, if there is no malicious adversary, then t crash failures have no effect on termination.) The $\Theta(t)$ factor is established by a lemma, interesting in its own right, that maintains the indistinguishability of two univalent configurations for t rounds. The 2β factor follows by reducing the 3-player game to consensus starting from these configurations—highlighting the utility of this technique even in the derivation of complicated, multi-part bounds.

Finally, in Section 7, we present tight upper bounds for reliable broadcast and consensus and approximately tight upper bounds for leader election and static k-selection.

[2]This bound may be surprising for two reasons: (1) it depends on n, even though $k < n$ players are contending, and (2) a larger k value results in a potentially faster protocol. For example, for $k = n - 1$, the problem can be solved by a single reliable broadcast by player 1, indicating whether she is a contender.

2 Related Work

This paper is the first to explore, in the context of a wireless network, a genuinely malicious (Byzantine) device that can reliably disrupt communication. Koo [9], Bhandari and Vaidya [3], as well as Pelc and Peleg [13] study "t-locally bounded" Byzantine failures in wireless networks, in which the number of Byzantine nodes in a region is bounded. In these papers, however, the Byzantine devices are required to follow a strict TDMA schedule, thus preventing them from interfering with honest communication. Drabkin et al. [8] allow Byzantine nodes to interfere with communication; each honest message, however, is successfully delivered with some non-zero probability, and authentication is available via public-key cryptography. Pelc and Peleg [14] also assume that messages may be corrupted (or lost) with some non-zero probability. By contrast, we consider an adversary that can choose to send a message in any round, potentially destroying honest messages or overwhelming them with malicious data. We assume no message authentication.

The basic model of communication on a shared medium goes back to research on Ethernet [1, 12]. For *wireless* networks, two issues are of particular importance: how the medium reacts to contention, and whether a device can detect contention/collisions. The subtlety of the first issue is illustrated in the contrasting results of Bar-Yehuda et al. [2] and Kowalski and Pelc [11]. If more than one message is sent, Bar-Yehuda et al. assume that a node may receive zero *or more* of the messages[3], while Kowalski and Pelc assume that a node receives exactly zero. In the former model, there is a linear-time lower bound for multi-hop broadcast which does not hold in the latter model. We assume the weaker communication model of Bar-Yehuda et. al., as it better captures the reality of radio broadcast behavior [16]. The second issue—collision detection—has two common variants: (1) a *CD model* where nodes can distinguish a silent round from a collision (e.g., [4, 7]); and (2) a *No-CD model* where nodes cannot distinguish these (e.g., [2, 11]). We assume the former model, which is realizable with existing carrier sensing mechanisms (e.g., [6, 15]).

3 Preliminaries

We now specify the details of our communication model. We assume a variant of the general Multiple Access Channel (MAC) model which is commonly used to represent single-hop radio network communication. We consider n honest devices, *the players*, each assigned a different label from the set $[1, n]$, and one additional malicious device incarnating *the adversary*. The players execute the algorithm assigned to

[3]There was a mistake in the original description of the model, as explained in the errata.

them, whereas the malicious adversary can deviate arbitrarily from the algorithm. Communication rounds are synchronous. In each round, each device can decide to broadcast a message or listen. For a given round, if there is no broadcast, then all players detect silence. If there is exactly one message broadcast, then all players receive the message. If two or more messages are broadcast, then each player can either: (1) receive exactly one of the broadcast messages; or (2) receive no messages, but detect noise on the channel, i.e., a collision. Without loss of generality, we assume that the adversary determines, for each honest player, whether option 1 or 2 occurs and, in case of option 1, which message is received. The adversary runs on a tiny device and therefore has a limited budget of β messages to broadcast, where β is *a priori* unknown to the players. Finally, we assume no message authentication capabilities. That is, a player cannot necessarily distinguish between a message sent from the adversary and a message sent by a fellow player.

The 3-Player Communication Game. The game involves two honest players and an adversary. For convenience, we refer to the honest players as Alice and Bob, and the adversary as Collin (the *Collider*). Initially, Alice is initialized with value $v_a \in V$ and Bob with $v_b \in V$, where $|V| > 1$ and V is known to both players. Each player attempts to communicate their initial value to the other. More formally, players can $\mathsf{output}(v)$ for any $v \in V$, and they must guarantee the following properties. **Safety:** Bob only outputs v_a and Alice only outputs v_b; and **Liveness:** Eventually, either Alice or Bob outputs a value.

4 Lower Bound

In this section we prove a lower bound on the termination time of the 3-player communication game. To do so, we describe a strategy for Collin to frugally use his β messages to prevent communication. Two assumptions are key to this strategy: (1) Collin's budget of messages β is unknown to Alice and Bob; (2) Alice and Bob cannot distinguish a message sent by Collin from an honest message. A *silent* round, on the other hand, cannot be faked: if Bob (for example) receives no message and no collision notification, then he can be certain that Alice did not broadcast a message. Therefore, in order to prevent Alice and Bob from communicating, it is sufficient, roughly speaking, for Collin to disturb silence.

Theorem 1. *Any 3-player communication protocol requires at least* $2\beta + \lg|V|/2$ *rounds to terminate, in the worst case.*

Assume, for contradiction, a protocol, A, that defies this worst-case performance. Consider any value $v \in V$ and denote by $\gamma(v)$ the $\lg|V|/2 - 1$ round (good) execution of A where Alice and Bob begin with initial

Rule	Alice		Bob		Collin	
	$\alpha(v)$	$\alpha(w)$	$\alpha(v)$	$\alpha(w)$	$\alpha(v)$	$\alpha(w)$
# 1	m	-	-	-	-	m
	-	m	-	-	m	-
# 2	-	-	m	-	-	m
	-	-	-	m	m	-
# 3	m	-	m'	-	-	m'
# 4	-	m	-	m'	m	

(a) α Rules

Rule	Alice		Bob		Collin
	$\alpha(v)$	$\rho(w,v)$	$\alpha(w)$	$\rho(w,v)$	$\rho(w,v)$
# 1	m	m'	-	-	m
# 2	-	-	m	m'	m

(b) $\rho(w,v)$ Rules

Figure 1: Collin's behavioral rules for α and $\rho(w,v)$ executions.

value v, and Collin performs no broadcasts. If Alice and Bob both broadcast in the same round, assume both messages are lost. We begin with the following lemma:

Lemma 2. *There exist two values $v, w \in V$ ($v \neq w$), such that Alice (resp. Bob) broadcasts in round r of $\gamma(v)$ if and only if Alice (resp. Bob) broadcasts in round r of $\gamma(w)$.*

Proof. In each round, there are four possibilities: (1) Alice broadcasts alone, (2) Bob broadcasts alone, (3) Alice and Bob both broadcast, and (4) neither Alice nor Bob broadcasts. Accordingly, for a sequence of c rounds, there are 4^c possible patterns of broadcast behavior. Thus, there are at most $4^{\lg|V|/2-1} = \frac{|V|}{4}$ possible broadcast patterns that result from the $|V|$ possible γ executions. It follows by the pigeonhole principle that at least two such executions have the same pattern. □

For the rest of this proof, we fix v and w to be the two values identified by Lemma 2. We define $\alpha(v)$ (resp. $\alpha(w)$) to be the execution of A in which Alice and Bob both begin with initial value v (resp. w) and Collin applies the α-rules described in Figure 1(a). In this table, "-" indicates silence and m and m' both represent a message broadcast. Each row matches a specific set of broadcast behaviors of Alice and Bob in two executions, with the corresponding broadcast behavior followed by Collin in these executions.

For example, Rule 1 from Figure 1(a) specifies that for any given round, if Alice broadcasts in exactly one α execution, and Bob is silent in both, then Collin replicates Alice's broadcast in the execution where she is silent. For any patterns of broadcast behavior not described in the table, assume that Collin performs no broadcasts. Also assume that in any round where both Collin and Alice (resp. Bob) broadcast, only Collin's message is received by Bob (resp. Alice). We claim the following:

Lemma 3. *Neither Alice nor Bob can output during $\alpha(v)$ or $\alpha(w)$.*

Proof. We show that Bob cannot output in $\alpha(v)$ and Alice cannot output in $\alpha(w)$. The argument for Bob in $\alpha(w)$ and Alice in $\alpha(v)$ is symmetric.

We first define a third execution $\rho(w, v)$, of A, in which Alice starts with initial value w and Bob starts with initial value v. Notice that, in all three of these executions, we assume that Collin has an unlimited broadcast budget. This is without loss of generality because Alice and Bob do not know the value of β, and we will later concern ourselves only with the prefixes of the α executions in which Bob has not yet broadcast more then β times.

We show, by induction on the round number, r, that $\rho(w, v)$ is indistinguishable from $\alpha(v)$ with respect to Bob, and that $\rho(w, v)$ is indistinguishable from $\alpha(w)$ with respect to Alice. The correctness of the lemma statement will follow immediately from this indistinguishability and the safety requirement of the communication game.

Since Bob begins with value v in both $\rho(w, v)$ and $\alpha(v)$, and Alice begins with value w in both $\rho(w, v)$ and $\alpha(w)$, the base case ($r = 0$) is immediate. We now consider the possible behaviors of Alice and Bob during round $r + 1$. We consider two symmetric cases.

- **Case 1:** *Alice broadcasts in* $\alpha(w)$. By induction, this implies Alice also broadcasts in $\rho(w, v)$, therefore these two executions remain indistinguishable with respect to Alice (as broadcasters cannot listen). Accordingly, we turn our attention to Bob. We bypass the sub-case of Bob broadcasting in $\alpha(v)$ as this is handled by Case 2. This leaves two possibilities. (1) Alice is silent in $\alpha(v)$. By α-rule 1 or 4, Collin will broadcast Alice's $\alpha(w)$ (and $\rho(w, v)$) message in $\alpha(v)$; (2) Alice broadcasts in $\alpha(v)$. By ρ-rule 1, Collin will replicate Alice's $\alpha(v)$ message in the ρ execution.

  Collin's behavior in response to both possibilities ensures that Bob receives the same message in $\alpha(v)$ and $\rho(w, v)$—providing the desired indistinguishability with respect to Bob.
- **Case 2:** *Bob broadcasts in* $\alpha(v)$. This argument is symmetric to Case 1. □

We now show that one of these two indistinguishable α executions requires only β broadcasts by Collin during the first $2\beta + \lg |V|/2 - 1$ rounds.

Proof (Theorem 1). By Lemma 3, Alice and Bob do not produce an output in either $\alpha(v)$ or $\alpha(w)$ as long as Collin continues to follow the α rules. It suffices to show that, in at least one of the two executions $\alpha(v)$ and $\alpha(w)$, Collin broadcasts in no more than β of the first $2\beta + \lg|V|/2 - 1$ rounds.

We first consider rounds 1 through $\lg|V|/2 - 1$ of both α executions. We know by Lemma 2 that Alice and Bob broadcast on the same schedule for these initial rounds when they both start with v and when they both start with w. Notice also, that the α rules only have Collin broadcast in situations of asymmetric silence (i.e. when Alice and Bob are *not* on the same schedule). It follows that Collin does not broadcast in either $\alpha(v)$ or $\alpha(w)$ for the first $\lg|V|/2 - 1$ rounds.

Now we turn our attention to the 2β rounds that follow. So far, Collin has expended no messages in either $\alpha(v)$ or $\alpha(w)$. Looking more closely at the α rules, however, it is clear that for a given round, Collin only broadcasts in $\alpha(v)$ or $\alpha(w)$, but not both, since he only fills in asymmetric silent rounds. Therefore, by a simple counting argument, it is impossible for Collin to broadcast in more than half of the rounds in both executions. We therefore choose the execution in which Collin broadcasts in no more than half of the following 2β rounds, as in this execution, Collin, using no more than β broadcasts, and yet delays both Alice and Bob from outputting for $2\beta + \lg|V|/2 - 1$ rounds. Contradicting our assumption about A. □

5 Upper Bound

In this section, we show that our lower bound for the 3-player game is tight by presenting a protocol that matches it. In fact, the protocol we describe solves a stronger version of the game in which Alice and Bob do not necessarily start at the same round, and Alice always succeeds in transmitting her value to Bob. Specifically, we assume that Alice is always awake, but does not receive her initial value v_a until some round r_a. Bob wakes up (and thus can start receiving messages) at some round r_b. The *duration* of the protocol is $r_{out} - \max(r_a, r_b)$, where r_{out} is the round during which Bob first outputs Alice's value. The presented protocol has a duration of $2\beta + O(\lg|V|)$ rounds, matching our lower bound.

Our protocol contains two sub-protocols: the *basic* sub-protocol and the *synchronization* sub-protocol. The basic sub-protocol assumes that Alice and Bob have already synchronized—they are starting the sub-protocol at the same round—and achieves the transmission of a single bit of information. The synchronization sub-protocol compensates for Alice and Bob's different starting rounds, and is used to help them start the basic sub-protocol at the same time.

Algorithm 1: Basic Sub-Protocol

```
1  basic-Alice(v, count)
2    if (count = 0) then return error
3    if (v='1') then bcast(v)    ▷ Data round broadcast
4    m ← recv()    ▷ Data round receive
5    if (m ≠ ⊥) then
6      count ← count-1
7      if (count = 0) then return error
8      if (v = '0') then bcast(veto)    ▷ Veto round broadcast
9    else if (m = ⊥) then
10     count ← 5lg |V|
11   m ← recv()    ▷ Veto round receive
12   if (m ≠ ⊥) then
13     return basic-Alice(v, count-1)
14   return ok
```

```
1  basic-Bob(count)
2    if (count = 0) then return error
3    m ← recv()    ▷ Data round receive
4    if (m = ⊥) then
5      estimate ← '0'
6      count ← 5lg |V|
7    else
8      estimate ← '1'
9      count ← count-1
10   if (count = 0) then return error
11   m ← recv()    ▷ Veto round receive
12   if (m = ⊥) then
13     return(estimate)
14   else
15     return basic-Bob(count-1)
```

Algorithm 2: Synchronization Sub-Protocol

```
1  synch(n)
2    if (n > 0) then
3      bcast(synch)
4      synch(n-1)
5    else synch-wait(0, 0)
```

```
1  synch-wait(n, max)
2    m ← recv()
3    if (m = ⊥) then
4      if (n = 0) then return
5      else synch-wait(max, max)    ▷ Restart wait
6    else if (m ≠ ⊥)
7      if (n > 0) then synch-wait(n-1, max)
8      else synch-wait(n, max)
```

Algorithm 3: Overall Communication Protocol

```
1  Alice()    ▷ Background process, preempted by bcast-Alice
2    repeat
3      synch-wait(5lg |V|, 5lg |V|)
4      basic-Alice('0')    ▷ False synchronization
5
6  bcast-Alice(value)
7    synch(5lg |V|)
8    bit ← -1
9    while (bit ≤ lg |V|) do
10     if (bit = -1) then    ▷ Real synchronization
11       status ← basic-Alice('1', 5lg |V|)
12       if (status = error) then synch-wait(0, 0)
13       else bit ← bit+1
14     else if (bit ≥ 0) then
15       status ← basic-Alice(value[bit], 5lg |V|)
16       if (status = error) then    ▷ Collin faked synch
17         synch-wait(0, 0)
18         bit ← -1
19       else bit ← bit+1
20       if (bit > lg |V|) then bcast-Alice(value)    ▷ Restart
```

```
1  recv-Bob()
2    synch-wait(5lg |V|)
3    bit ← -1
4    while (bit ≤ lg |V|) do
5      v ← basic-Bob(5lg |V|)
6      if (v = error) then
7        synch-wait(0, 0)
8        bit ← -1
9      else if (bit = -1) then
10       if (v = '0') then synch-wait(5lg |V|)
11       else if (v = '1') then bit ← bit+1
12     else if (bit ≥ 0) then
13       value[bit] ← v
14   return value
```

The Basic Sub-Protocol. The pseudo-code for the basic sub-protocol is depicted in Algorithm 1 and it begins with a call by Alice to basic-Alice(v), where v is a single bit to be transmitted, and a call by Bob to basic-Bob() which returns a single bit.

The sub-protocol alternates data rounds and veto rounds. In a data round, Alice transmits a message if her initial value is '1', and remains silent otherwise. If Alice's initial value is '0', but Collin broadcasts during the data round, then Alice broadcasts in the veto round, preventing Collin from confusing Bob. If Bob detects silence in the data round and silence in the veto round, then Bob can safely accept '0' as the value that Alice was trying to transmit. Alternatively, if Bob receives a message in the data round and detects silence in the veto round, then Bob can safely accept '1'. In both cases, both Alice and Bob are aware that the value has been successfully transmitted, and can both therefore terminate the basic sub-protocol at the same round. If, on the other hand, Bob receives a message or detects a collision in the veto round, then both Alice and Bob begin again with another pair of data and veto rounds.

The Synchronization Sub-Protocol. The synchronization sub-protocol (Algorithm 2) ensures that Alice and Bob begin the basic sub-protocol in the same round, despite potentially waking up in different rounds. Synchronization occurs when Alice (or Collin) broadcast for at least $5 \lg |V|$ rounds, followed by a single round of silence. The synch-check procedures listens for such a sequence of rounds. The synch procedure is used by Alice to produce such a sequence of rounds.

Roughly speaking, the large number of messages required to complete the synchronization protocol is meant to discourage Collin from forcing Alice and Bob to restart in the middle of data transmission, as it would be a waste of his limited broadcast budget.

The Overall Protocol. The overall protocol (see Algorithm 3) has three stages: (1) *Preliminary*: In the beginning, Alice has not yet learned the value to broadcast but needs to prevent Collin from pretending to Bob that she has. (2) *Synch*: As soon as the bcast-Alice(v) function is called, notifying Alice of her value, the synchronization sub-protocol begins. (3) *Basic*: When synchronization finishes, then the basic sub-protocol begins, transmitting each bit in the initial value. When the basic sub-protocol completes, the overall protocol returns to the second stage, performing the synchronization sub-protocol again.

In the preliminary stage, Alice has not yet been initialized with her value, and hence cannot transmit the value to Bob. Nevertheless, Alice must ensure that Collin is not tricking Bob into receiving the wrong value.

Therefore, Alice uses the synchronization protocol's synch-check functionality to monitor whether Collin has performed a synch, broadcasting in $5 \lg |V|$ consecutive rounds, followed by silence. In this case, Bob may be preparing to begin the basic protocol, thinking that Alice is about to begin. Alice therefore invokes the basic sub-protocol to transmit '0' to Bob, which Bob interprets as indicating a fake synchronization.

As soon as Alice is provided a value to broadcast, she begins the actual synchronization sub-protocol. She initializes it with a large constant, $5 \lg |V|$, causing a large number of consecutive rounds of messages, followed by one round of silence. If Collin prolongs the synchronization by broadcasting during the final round of silence, then the protocol terminates in the first silent round.

As soon as the synchronization sub-protocol completes, Alice begins the basic sub-protocol. Alice invokes the basic sub-protocol $\lg |V| + 1$ times. First, Alice invokes the basic sub-protocol transmitting '1'. This indicates to Bob that the synchronization was valid, and is not a trick by Collin. Then, Alice iterates through each bit in the value to be broadcast, transmitting it to Bob.

At anytime during the basic sub-protocol, Collin may choose to broadcast in a large number of consecutive rounds, forcing resynchronization. In this case, Alice must start over with the first bit of her value. Otherwise, it's possible that Bob wakes up just as Collin's broadcasts begin, causing him to think that data transmission is just beginning even though Alice plans to continue after Collin's fake synchronization where she left off. This lack of synchronization could lead to a safety violation.

Once Alice completes sending all the bits, she returns to the synchronization stage; she does not know whether Bob has awakened (and received the value) or if he is still asleep. Bob, on the other hand, can output the value and terminate as soon as he receives all the bits in Alice's value.

Theorem 4 (Safety). *Bob does not output any value that is not Alice's initial value.*

Proof (Sketch). First consider sending a single bit using the basic sub-protocol, assuming Alice and Bob begin in the same round. If Alice begins with value '1', then she broadcasts in every data round, and hence Bob will never output '0'. On the other hand, if Alice begins with value '0', then she broadcasts in every veto round which is preceded by a broadcast by Collin, and hence Bob will not output a '1'. Moreover, as described previously, Alice and Bob both conclude the basic sub-protocol in the same round.

Next, we argue that Alice and Bob repeatedly begin the basic sub-protocol in the same round. Alice continually monitors the channel for a sequence of $5 \lg |V|$ non-silent rounds, followed by a silent round,

which indicates a restart of the data transmission. After any such sequence, Alice uses the basic sub-protocol to broadcast a zero or a one; the correctness of the basic sub-protocol (above) indicates that Bob will correctly determine whether to begin receiving Alice's value, or wait for a future synchronization. Hence Alice and Bob begin the basic protocol for the first bit—and all subsequent bits—in the same round. □

Theorem 5 (Liveness). *If Alice receives the value to broadcast in round r_a, and Bob wakes up in round r_b, then Bob outputs Alice's initial value no later than round* $\max(r_a, r_b) + 2\beta + 14 \lg |V| + 5$.

Proof (sketch). With no delays, Alice and Bob will finish in $\max(r_a, r_b) + 14 \lg |V| + 5$: if $r_b = r_a + 5 \lg |V|$, then Bob just misses the beginning of Alice's synchronization, resulting in two iterations of synchronization ($5 \lg |V| + 3$), and two iterations of bit transmission ($2 \lg |V|$), less the round Bob arrived too late. Collin can delay each of the iterations of the basic protocol by 2 rounds for each broadcast, resulting in an additive 2β.

Note that Collin gains no advantage by disrupting synchronization. The best strategy for Collin to disrupt synchronization is to perform $5 \lg |V|/2$ broadcasts (alternating with Alice attempting to send a '1') and thus forcing a reset of data transmission. This delays Alice and Bob for $2 \lg |V| + 3$ additional rounds. Collin only gains, on average, less than two rounds of delay for every malicious broadcast. □

6 n-Player Lower Bounds

This section supports our claim that the misadventures of Alice, Bob, and Collin capture something fundamental about the ability of a malicious device to disrupt coordination in a single-hop wireless network. We first study the impact of a malicious device (which can itself be representing a set of malicious devices) on n honest devices. We then consider the possibility that a subset of these n honest devices might crash and study the impact of combining malicious behavior with these failures.

6.1 n-Player Simulation

We show here how Alice and Bob can together simulate an arbitrary n-player protocol. We then use this simulation to derive lower bounds, via reduction from the 3-player communication game, for several n-player problems: reliable broadcast, leader election, static k-selection, and, later, consensus.

A simulation by Alice and Bob is defined by a 5-tuple: $\{A, n, S_A, S_B, I\}$, where: (1) A is the n-player protocol being simulated; (2) S_A and S_B partition the n players into two non-empty and non-overlapping sets; (3) I is a mapping of players to their respective initial values. Alice and Bob simulate the players in S_A and S_B, respectively. Accordingly, we assume that Alice is provided only the initial values for nodes in S_A (i.e., $I|S_A$) and Bob is provided only the initial values for nodes in S_B (i.e., $I|S_B$).

We describe Alice's simulation protocol. (Bob's is equivalent.) Alice simulates the players in S_A, initializing them according to I. Each simulated round r consists of two parts: broadcasting and receiving. *Broadcasting:* If any of the players in S_A broadcast in round r (according to A), then Alice arbitrarily chooses one message and broadcasts it. (The remaining messages are ignored.) *Receiving:* If Alice broadcast a message m, then she delivers m to all her simulated players. Otherwise, she simulates each player either receiving a message, detecting silence, or detecting a collision—depending on what Alice herself detected. Finally, Alice outputs the actions of her simulated players.

Theorem 6. *Consider simulation $\{A, n, S_A, S_B, I\}$. For all r-round executions of the simulation, there exists an r-round execution α of A, initialized according to I, where the outputs of Alice and Bob are equivalent to the outputs in α, and Collin broadcasts the same number of messages in the simulation and α.*

Proof (Sketch). We prove this claim by a straightforward induction on the round number, showing that after r' rounds: (1) the state of the simulated players corresponds to some r'-round legal execution of A; and (2) Collin has performed the same number of broadcasts in both the simulation and the execution of A.

There are two cases of interest. First, consider the case where two or more of Alice's simulated players broadcast (resp. Bob's players), and the simulation algorithm has Alice (resp. Bob) choose only one message to broadcast. In our α execution, this matches the case in which a single message overwhelms others broadcast in the same round. Another interesting case occurs when Collin broadcasts in the simulation. In our α execution, this matches the case where Collin broadcasts the same message, overwhelming other messages and/or causing collisions in the same pattern seen in the simulator. This is the only case in which Collin broadcasts in the α execution; preserving the second property of our hypothesis. □

Reliable Broadcast. In reliable broadcast, one player—the *source*—is provided with an input value $v_0 \in V$. Each player must receive this initial value. *Safety* requires that each player output only v_0, i.e., perform

output(v) only if $v = v_0$. *Liveness* requires that all players eventually perform an output. Once all players have performed an output, we say the protocol has *terminated*. We establish the following lower bound:

Theorem 7. *Any reliable broadcast protocol requires at least* $2\beta + \lg|V|/2$ *rounds to terminate, in the worst case.*

Proof. Assume by contradiction that A is a reliable broadcast protocol that terminates in $R < 2\beta + \lg|V|/2$ rounds for all initial values. We reduce 3-player communication, for value domain V, to A. Alice and Bob simulate A for n players, where: (1) S_A contains the source, S_B contains all other players, and (2) I maps the source to v_a, Alice's initial value. Bob outputs the first value output by a simulated player. By Theorem 6, Bob always outputs $v_0 = v_a$ by round R, contradicting Theorem 1. □

Leader Election. In leader election, $k \leq n$ *participants* contend to become the leader. All n players should learn the leader, i.e., perform output(ℓ), for some ℓ. *Safety* requires that the leader be a participant, and that there be only one leader. *Liveness* requires every player to perform an output. The protocol *terminates* when every player has performed an output. We establish the following lower bound:

Theorem 8. *Any leader election protocol requires at least* $2\beta + \lg\lfloor\frac{n-1}{k}\rfloor/2$ *rounds to terminate, in the worst case.*

Proof. Assume by contradiction that A is a leader election protocol that terminates in $R < 2\beta + \lg\lfloor\frac{n-1}{k}\rfloor/2$ rounds for all choices of k participants. We reduce to leader election, the 3-player game defined over the value space V, where V contains every integer between 1 and $\lfloor\frac{n-1}{k}\rfloor$, to A.

Alice and Bob simulate A for n players where: (1) S_A contains players 1 through $n-1$, S_B contains player n, and (2) I activates player $i \in S_A$ if and only if $(i \ mod \ \lfloor\frac{n-1}{k}\rfloor) + 1 = v_a$ and fewer than k nodes have been activated so far in I_A. Let i be the leader output by Bob's simulated player. Bob outputs $v_a = (i \ mod \ \lfloor\frac{n-1}{k}\rfloor) + 1$, as required. By Theorem 6 Bob always outputs v_a within R rounds, contradicting Theorem 1, since $2\beta + \lg V/2 = 2\beta + \lg\lfloor\frac{n-1}{k}\rfloor/2$. □

Static k-Selection. In static k-Selection, k *participants* are provided with values $v_i \in V$. Each player must receive of of these values. *Safety* requires that the first k outputs of a player equal the k initial values. *Liveness* requires that all players eventually perform at least k output actions. The protocol *terminates* when

all players have performed at least k output actions. (Note: the selection problem is well-studied in radio networks, e.g., [5, 10].) We establish the following lower bound:

Theorem 9. *Any static k-selection protocol requires at least $2\beta + \Omega(k \lg \frac{|V|}{k})$ rounds to terminate, in the worst case.*

Proof. Assume by contradiction that A is a protocol that terminates in $R < 2\beta + o(k \lg |V|/k)$ rounds, for all initial values and choices of participants. We reduce to k-selection, the 3-player game for the value space V', where V' contains one entry for every multiset of k values drawn from V, to A.

Alice and Bob simulate A for n players where: (1) S_A contains players 1 through k, S_B contains the remaining players, and (2) I activates players 1 through k, and provides each a different value from the multiset described by $v_a \in V'$. Given k simulated outputs, Bob can reconstruct and output the unique multiset described by these values. By Theorem 6 Bob will always output v_a in R rounds, contradicting Theorem 1, since $2\beta + \lg |V'|/2 = 2\beta + \lg \frac{|V|^k}{k!}/2 = 2\beta + \Theta(k \lg \frac{|V|}{k})$ rounds. □

6.2 Combining Malicious and Crash Behavior

We now study the impact of combining malicious behavior with crash failures. We assume that the adversary, in addition to controlling a malicious device with a budget of β messages, can also crash up to t players. We consider *binary consensus* as an archetypal problem in this context. In consensus, the n honest players each propose an initial value. *Liveness* requires that all non-crashed players eventually decide a value. *Agreement* requires all players that decide to choose the same value. *Validity* requires that if all non-crashed players propose the same value, then all players that decide choose that value.

By a simple indistinguishability argument, it is easy to see that consensus is impossible if $n \leq 2t$: it is impossible to distinguish a correct player from a crashed player that is simulated by the adversary; thus no player can decide in an execution in which t players propose '0' and t propose '1'. We assume that $n > 2t$.

In this section, we establish a lower bound of $2\beta + \Theta(t)$ on the round complexity of consensus. Our bound reveals the interesting fact that the possibility of crashed honest devices increases the power of the malicious adversary. This is perhaps surprising as, if there is no malicious adversary, crash-failures have no effect on termination (in a synchronous broadcast network).

As before, we use a simulation by Alice and Bob of the (t-resilient) n-player consensus protocol. The simulation, however, is more challenging than those used for the n-player problems we studied so far. This

is because we must compensate for the crash failures. We do not start the simulation from the initial configuration, but instead from one of two univalent configurations arising after t rounds of computation. These configurations are constructed in Lemma 11, which is interesting in its own right as it exhibits executions in which information (about initial values) is transmitted at most one bit per round. By combining it with valency arguments, we show how the 3-player game can aid the construction of involved lower bounds.

Theorem 10. *Any t-resilient binary consensus protocol requires at least $2\beta + t$ rounds, in the worst case.*

Assume $n = 2t + 1$. We fix the environment such that if multiple messages are sent in a round, and the adversary does not broadcast, then the message sent by the player with the smallest id is received by everyone. An execution (or prefix) is *failure-free* if it includes no crashes or broadcasts by the adversary.

Given these assumptions, it is clear that each initial configuration results in a deterministic failure-free execution. We represent all of these possible failure-free executions as a tree T. Every execution begins at the root, and a node at depth r represents the execution at the beginning of round r. Each node at depth r contains one outgoing edge for every possible message m that may be received in round r, and one outgoing edge for a silent round (labeled $\perp$). Thus, every failure-free execution of A is represented by a single path in T. Accordingly, for each initial configuration c, we say that a node $x \in T$ is *reachable* from c—with respect to A—if the path associated with c's failure-free execution includes node x. We define the tree $T(A)$ to be T pruned to contain only reachable nodes. That is, if $x \in T(A)$, then there exists some initial configuration c for which x is reachable. Notice that if a depth r node x is reachable for two initial configurations c and c', and some player i has the same initial value in c and c', then at the beginning of round r, player i cannot distinguish c from c'. If c is 0-valent, and c' is 1-valent, then i cannot decide prior to round r.

Lemma 11. *There exists a path of length t in $T(A)$, ending at node R_t, where R_t is reachable from two initial configurations, c_0 and c_1, such that some player p_t has the same initial value in c_0 and c_1, and every crash-free extension of c_0 is 0-valent and every crash-free extension of c_1 is 1-valent, with respect to A.*

Proof. Starting at the root of $T(A)$, we iteratively construct a path of length t by applying the following rules: (1) If there exists ≥ 1 outgoing message edges, choose the message from the player with the smallest id. (2) Otherwise, follow the $\perp$ edge. Let R_t be the node reached after t iterations of this rule.

Let c_0 be some configuration from which R_t is reachable. Since $n = 2t + 1$, configuration c_0 contains either a majority of initial '0's or a majority of '1's. Without loss of generality, assume that at least $t + 1$

players propose '0' in c_0. This implies that any crash-free extension of c_0 must decide '0', since any such execution is indistinguishable from one in which all players propose '0', and those $< t$ players proposing '1' are crashed nodes emulated by the adversary—a case in which a decision of '1' would violate validity.

We now construct an initial configuration c_1. Denote by P the set of players that broadcast messages which were received along the path to R_t. Note that P contains $\leq t$ players. Choose c_1 such that the players in P propose the same initial value as in c_0, and the remaining players (at least $t+1$) all propose '1'. Choose some $p_t \in P$. (If $|P| = 0$, then arbitrarily choose one player p_t to have the same initial value in c_0 and c_1.) It is clear, by the same reasoning applied to c_0, that all crash-free extensions of c_1 must decide '1'.

We claim that R_t is reachable from c_1. This follows by a straightforward induction. The base case (the root) is clear. Our hypothesis posits a node $R_{t'}$, on the path to R_t, such that $R_{t'}$ is reachable from c_1. Let e be the outgoing edge from $R_{t'}$ on the path to R_t. There are two cases. (1) Edge e is associated with a message m broadcast by some player $i \in P$. Since player i begins with the same initial value in both c_0 and c_1, and has seen the same sequence of messages and silence up to this point, it broadcasts the same message m in the failure-free execution generated by c_1. No player with a smaller id can also broadcast in this round of the c_1 execution as this contradicts the path construction. (2) Edge e is labeled $\perp$. By the path construction, $\perp$ is the only outgoing edge from $R_{t'}$, thus it must be followed by the c_1 execution. □

Proof (Theorem 10). Let α_0 (resp. α_1) denote the failure-free consensus execution prefix starting from c_0 (resp. c_1) and described ending at R_t. Notice that α_0 and α_1 are indistinguishable with respect to p_t, and hence p_t has not decided prior to round t. To this point, the adversary has used zero broadcasts from his budget. To achieve a further 2β delay, we defer, as always, to our helpful friends: Alice, Bob, and Collin.

Assume for contradiction that the consensus protocol always terminates in less than 2β rounds starting from the end of α_0 and α_1. We reduce binary 3-player communication to a consensus execution starting from these states. Alice and Bob perform a *crash-free* simulation of the protocol for n players, where: (1) S_A contains all players except p_t, and S_B contains p_t, and (2) I maps players to the state described by α_0, if $v_a = 0$, or α_1, if $v_a = 1$. (Player p_t is mapped to the same state in both cases, since the executions are indistinguishable to p_t to this point.) Bob outputs the decision of p_t. By Theorem 6, Bob always outputs v_a within 2β rounds, contradicting Theorem 1. □

7 n-Player Protocols

In this section, we discuss protocols for reliable broadcast, leader election, static k-selection, and consensus in n-player networks. Our reliable broadcast and consensus protocols match our lower bounds. Those for leader election and k-selection do not—implying interesting open research directions.

Reliable Broadcast. An algorithm for reliable broadcast follows immediately from the algorithm presented in Section 5. In particular, the source runs Alice's protocol, and all other players run Bob's protocol, resulting in a running time of $2\beta + O(\lg |V|)$, matching the lower bound.

Consensus. Assuming t crashes (besides the malicious adversary), a consensus protocol can be obtained by using the reliable broadcast protocol for each of $2t + 1$ players to transmit their initial value sequentially. (Notice that a crashed player, if there is no malicious interference, transmits a '0', according to the protocol.) At this point, everyone can safely decide the majority value. The running time is $2\beta + \Theta(t)$.

Leader Election. We now briefly sketch a leader election protocol that uses a tournament tree to select one participant as leader. The protocol is parameterized by a constant integer $c \geq 1$. Consider a binary tree with n leaves, each labeled with the id of a single player. The protocol begins at the root of the tree, and at each step descends either to the left or right sub-child, or ascends to the parent. On reaching the leaf of the tree—and ensuring that the leaf node is a valid participant—the protocol terminates. Each step of the tree walk consists of two c-round phases—the left-child phase and the right-child phase. In the left-child phase, every participating player identified with a leaf of the left subtree broadcasts. Conversely, players in the right subtree broadcast during the right-child phase. A phase ends after the first silent round, or after c non-silent rounds. In the latter case, we say that the phase was *successful*. If the left-child phase was successful, the protocol descends to the left; otherwise, if the right-child phase was successful, the protocol descends to the right; if neither round was successful, the protocol ascends to the parent. If there is no parent—because the current tree node is the root—then there are no participating players. If there is no malicious interference, the protocol reaches a leaf in $2c \lg n$ rounds.

On reaching a leaf, the protocol ensures that the identified player is a participant—not simply a product of malicious interference. To achieve this, the identified player uses reliable broadcast to transmit a '1' if she is participating, and a '0' otherwise. In the latter case, the protocol ascends to the parent and continues.

Theorem 12. *The leader election protocol terminates after* $2\beta\frac{c+1}{c} + 2c \lg n + 2$ *rounds, for all* $c \geq 1$.

Proof (sketch). We say an edge is *good* if some participant is in the subtree of that edge. At each of the $\lg n$ levels of the tree, the protocol traverses only one good edge, resulting in a cost of $2c \lg n$. The traversal of each bad edge results in at most $2c + 2$ rounds (amortized): $2c$ rounds to descend and 2 rounds to later ascend. With adv broadcasts, the adversary can cause the protocol to traverse at most β/c bad edges. □

k-Selection. A protocol for static k-selection can be obtained by repeating the leader election protocol k times, each time using reliable broadcast to transmit the initial value. The protocol completes when leader election finds no further contenders. Assuming $\lg n = O(\lg |V|)$, which is commonly the case:

Theorem 13. *The k-selection protocol terminates after $2\beta\frac{c+1}{c} + 2kc \lg n + k \lg V + 2k + 2$, which is equal to $2\beta\frac{c+1}{c} + O(ck \lg |V|)$ rounds, for all $c \geq 1$.*

References

[1] N. Abramson. The aloha system - another approach for computer communications. *Proceedings of Fall Joint Computer Conference, AFIPS*, 37:281–285, 1970.

[2] 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.

[3] Vartika Bhandari and Nitin H. Vaidya. On reliable broadcast in a radio network. In *Proceedings of the 24th ACM Symposium on Principles of Distributed Computing*, pages 138–147, 2005.

[4] Bogdan S. Chlebus, Leszek Gasieniec, Alan Gibbons, Andrzej Pelc, and Wojciech Rytter. Deterministic broadcasting in ad hoc radio networks. *Distributed Computing*, 15(1):27–38, 2002.

[5] Andrea E. F. Clementi, Angelo Monti, and Riccardo Silvestri. Selective families, superimposed codes, and broadcasting on unknown radio networks. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 709–718, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

[6] J. Deng, P. K. Varshney, and Z. J. Haas. A new backoff algorithm for the IEEE 802.11 distributed coordination function. In *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation*, January 2004.

[7] Anders Dessmark and Andrzej Pelc. Tradeoffs between knowledge and time of communication in geometric radio networks. In *Proceedings of the 13th ACM Symposium on Parallel Algorithms and Architectures*, pages 59–66, 2001.

[8] Vadim Drabkin, Roy Friedman, and Marc Segal. Efficient byzantine broadcast in wireless ad hoc networks. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 160–169, 2005.

[9] C-Y. Koo. Broadcast in radio networks tolerating byzantine adversarial behavior. In *Proceedings of the 23rd ACM Symposium on Principles of Distributed Computing*, pages 275–282, 2004.

[10] Dariusz R. Kowalski. On selection problem in radio networks. In *Proceedings of the twenty-fourth annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 158–166, New York, NY, USA, 2005. ACM Press.

[11] Dariusz R. Kowalski and Andrzej Pelc. Deterministic broadcasting time in radio networks of unknown topology. In *Proceedings of he 43rd IEEE Symposium on Foundations of Computer Science*, pages 63–72, 2002.

[12] R. M. Metcalf and D. R. Boggs. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM*, 19(7):395–404, July 1976.

[13] Andrzej Pelc and David Peleg. Broadcasting with locally bounded byzantine faults. *Information Processing Letters*, 93(3):109–115, 2005.

[14] Andrzej Pelc and David Peleg. Feasibility and complexity of broadcasting with random transmission failures. In *Proceedings of the 24th ACM Symposium on Principles of Distributed Computing*, pages 334–341, 2005.

[15] J. Polastre and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems*, pages 95–107, 2004.

[16] A. Woo, K. Whitehouse, F. Jiang, J. Polastre, and D. Culler. Exploiting the capture effect for collision detection and recovery. In *Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors*, pages 45–52, May 2005.