Leveraging Channel Diversity to Gain Efficiency and Robustness for Wireless Broadcast

Shlomi Dolev[1], Seth Gilbert[2], Majid Khabbazian[3], and Calvin Newport[4] *

[1] Ben-Gurion University, Beersheba, Israel
[2] National University of Singapore, Singapore
[3] University of Winnipeg, Winnipeg, Canada
[4] MIT CSAIL, Cambridge, USA

Abstract. This paper addresses two primary questions: (i) How much faster can we disseminate information in a large wireless network if we have multiple communication channels available (as compared to relying on only a single communication channel)? (ii) Can we still disseminate information reliably, even if some subset of the channels are disrupted? In answer to the first question, we reduce the cost of broadcast to $O(\log \log n)$ rounds/hop, approximately, for sufficiently many channels. We answer the second question in the affirmative, presenting two different algorithms, while at the same time proving a lower bound showing that disrupted channels have unavoidable costs.

1 Introduction

This paper addresses two primary questions: (i) How much faster can we disseminate information in a large wireless network if we have multiple communication channels available (as compared to relying on only a single communication channel)? (ii) Can we still disseminate information reliably, even if some subset of the channels are disrupted? In answer to the first question, we reduce the cost of broadcast to $O(\log \log n)$ rounds/hop, approximately, for sufficiently many channels. We answer the second question in the affirmative, presenting two different algorithms, while at the same time proving a lower bound showing that disrupted channels have unavoidable costs.

Multi-channel Networks. In more detail, we study the multihop broadcast problem in the t-disrupted radio network model [11–14, 17, 26, 30, 31]. This model describes a synchronous multihop radio network, and assumes that in each round, each process chooses 1 out of $\mathcal{C}$ available communication channels to participate on. Simultaneously, an adversary selects, at each process, up to $t < \mathcal{C}$ channels to locally *disrupt*, preventing communication. In this study, we also equip processes with receiver collision detectors, but assume that disruption is indistinguishable from collisions.

As detailed in [11, 26], the adversary in the t-disrupted model does not represent a literal adversarial device; it cannot spoof messages or reactively jam a broadcast (i.e.,

* Research supported in part by the ICT Programme of the European Union under contract number FP7-215270 (FRONTS), Microsoft, Deutsche Telekom, VeriSign .Com grant, Israeli Defense Secretary (MFAT), US Air-Force and Rita Altura Trust Chair in Computer Sciences, NUS (FRC) R-252-000-443-133, and Natural Sciences and Engineering Research Council of Canada.

scan the channels to discover a broadcast in progress, then jam the remainder of transmission). The adversary instead incarnates the unpredictable message loss that plagues real radio network deployments. This message loss has many (non-malicious) causes, including: unrelated protocols using the same unlicensed spectrum band, time-varying multipath effects, and electromagnetic interference from non-radio devices, such as microwaves. The goal of the t-disrupted model is two-fold: (1) to improve *efficiency*: most real radio network protocols have access to multiple communication channels,[5] and therefore theoretical algorithms should enjoy this same advantage; and (2) to improve *robustness*: a protocol proved correct in a model with unpredictable message loss is a protocol more likely to remain correct in a real deployment, where such loss is often unavoidable.

Results: No Disruption. We start by showing that adding communication channels makes broadcast more efficient, yielding $O(\log\log n)$ rounds/hop in a network of diameter $D > \log n$ with $\Theta(\log n)$ channels. In more detail, in the setting with no disruption ($t = 0$), we present a randomized algorithm that solves broadcast in $O((D + \log n)(\log \mathcal{C} + \frac{\log n}{\mathcal{C}}))$ rounds, w.h.p. Notice, for a single channel ($\mathcal{C} = 1$), our algorithm has the same running time as the canonical Bar-Yehuda et. al algorithm [3], but as the number of channels increases so does our algorithm's performance advantage. This comparison however, is not exact, as unlike [3], we assume collision detection. With this in mind, we prove a lower bound $\Omega(D + \frac{log^2 n}{\mathcal{C}})$ rounds for broadcast algorithms in our collision detector-equipped model. It follows that for $\mathcal{C} = \Omega(\log n)$: our algorithm is within a factor of $O(\log\log n)$ of optimal, and for sufficiently small D, it is strictly more efficient than the best possible single-channel algorithm.

The key insight of this algorithm is the following: At a high-level, standard single-channel broadcast algorithms, such as [3], require processes to sequentially test $\log n$ broadcast probabilities, exponentially distributed between $1/n$ and $1/2$. The idea is that for every process with transmitting neighbors, one of these probabilities will match what is required for the message to be received. Our algorithm, by contrast, leverages multiple communication channels to test multiple probabilities *in parallel*, allowing processes to hone in on the correct probabilities more efficiently.

While it may not be surprising that some speed-up is possible using multiple channels, it is non-trivial to determine exactly what is feasible for two reasons. First, the multiple communication channels can only speed up one part of the algorithm (i.e., the contention resolution); it cannot speed-up the time to relay the message over long distances. Second, the multiple channels cannot all be used in parallel by any one processor, as each has only one transceiver. Thus, the "obvious" solutions, e.g., multiplexing the single-channel protocol over multiple channels, are not applicable. If $\log n$ channels could be used in parallel, we could readily achieve a rounds-per-hop cost of $O(1)$; that we can still achieve $O(\log\log n)$ rounds-per-hop with only one transceiver, is, perhaps, surprising.

Results: t-Disruption. Having showing that additional communication channels improves *efficiency*, we next turn our attention to showing that they also improve *robustness*. We now assume disruption (i.e., $t > 0$) and that processes have access to a com-

[5] The 802.11 b/g network protocols [1], for example, divide the shared 2.4 Ghz band into 13 channels, while Bluetooth [4] divides the same band into 79 channels.

mon source of random bits. We argue that this latter assumption is often justified in practice, as most radio network deployments require devices to be configured with a common *network id*, and a hash of this id can provide a seed for a pseudo-random bit generator.[6]

In this setting, we present a randomized algorithm that solves broadcast in $O((D + \log n)(\frac{\mathcal{C}\log\mathcal{C}\log\log\mathcal{C}}{\mathcal{C}-t} + \frac{\log n}{\mathcal{C}-t}))$ rounds, w.h.p., where t is the upper bound on disrupted channels. Notice, for t up to a constant factor of $\mathcal{C}$, this algorithm performs only a factor of $O(\log\log\mathcal{C})$ slower than the no disruption case. In other words, even with lots of disruption, our multi-channel algorithm still outperforms the best possible single channel solution in many cases, and is more efficient than the canonical single channel algorithm of [3]. The key insight of this algorithm is that we replace the broadcast and receive primitives used in the no disruption case with *simulated* versions. These simulated broadcasts and receives use the common randomness to generate coordinated random frequency hopping patterns. These patterns are used to evade adversarial disruption with sufficient probability for the original no disruption arguments to still apply.

Lastly, we consider the case with disruption and *no* common randomness. We describe a randomized algorithm that solves broadcast in this setting in $O((D+\log n)\frac{\mathcal{C}t}{\mathcal{C}-t}\cdot\log(\frac{n}{t}))$ rounds, w.h.p. Notice, for large t, this algorithm now performs slightly worse than [3], but this is arguably still a reasonable price to pay for the added robustness. We conclude by showing this price to be not just reasonable, but also be necessary. In more detail, we prove a lower bound of $\Omega((D + \log n)\frac{\mathcal{C}t}{\mathcal{C}-t})$ rounds to solve broadcast in this setting.

Related Work. We use the terminology *multihop broadcast* to describe the problem addressed in this paper, as we want to clearly separate it from the *local broadcast* problem we solve as a subroutine. Previous work on this problem, however, has used both *reliable broadcast* (e.g., [19]) and *broadcast* (e.g., [3]) to refer to the same problem. All terms describe the same goal of disseminating a message from a single distinguished source to every process in a radio network.

Theoretical concern with broadcasting in radio networks began with the investigation of centralized solutions. Chlamtac and Kutten [5] opened the topic by proving the calculation of optimal broadcast schedules to be NP-hard, Chlamtac and Weinstein [6] followed with a polynomial-time algorithm that guaranteed schedule lengths of size $O(D\log^2 n)$, and Alon et al. proved the existence of constant diameter graphs that require $\Omega(\log^2 n)$ rounds [2]. An oft-cited paper by Bar Yehuda et al. [3] introduced the first *distributed* solution to broadcast, launching a long series of papers investigating distributed solutions under different model assumptions; c.f., [7–10, 22]. The algorithm in [3] assumes no topology knowledge or collision detection, and solves broadcast in $O((D + \log n)\log(n))$ rounds, w.h.p. In later work [10, 21], this bound was improved to $O((D+\log n)\log(n/D))$, which performs better in graphs with large diameters. For the assumption of no topology knowledge, these broadcast bounds can be considered the best known. In centralized setting, the optimal result $\Theta(D + \log^2 n)$ in undirected multi-hop networks is given in [16] and [23].

[6] Note, if the adversary in the t-disrupted model represented an actual adversarial device, we would have to worry about keeping such information secure. But as explained previously, this adversary is an abstraction of the diverse, and hard to predict interference sources that plague real networks, and does not represent behavior with malicious intent.

Our algorithm for the no disruption setting matches the Bar-Yehuda algorithm for the case where $\mathcal{C} = 1$, and performs increasingly better as we increase the number of channels. Its comparability with the bound of [10, 21] depends on the diameter. Our model, however, unlike the model in [3, 10, 21], assumes receiver collision detection, so these comparisons are not exact. (The $O(D \log n)$-time broadcast algorithm of [29], by contrast, *does* assume collision detection, but a direct comparison is foiled in this case because the model of [29] constrains the communication graph to be growth-bounded, whereas our model, as in the canonical results referenced above, works for arbitrary graphs.) This motivates the $\Omega(D + \frac{\log^2 n}{\mathcal{C}})$ lower bound we prove in Section 6 for solving broadcast in our model. Notice, this implies that the best possible single-channel broadcast algorithm in our model requires $\Omega(D + \log^2 n)$ rounds. For $\mathcal{C} = \Omega(\log n)$, and sufficiently small D, our no disruption algorithm is strictly more efficient. In Section 4, we show that even if we introduce significant disruption, if we assume a common source of randomness we still outperform the best possible single channel solution in many cases.

Koo [19] considered broadcast in a model that assumed a single channel and Byzantine failures, which, due to their ability to spoof messages, are arguably more challenging than the disruption faults considered in our work. The corrupt processes in this model, however, could not disrupt communication. In later work, Koo, now collaborating with Bhandari, Katz, and Vaidya [20], extended the model to allow for a bounded number of collisions. Their focus was on feasibility (i.e., for what amount of corruptions is broadcast still solvable) not time complexity. Drabkin et al. [15] and Pelc and Peleg [27] both studied broadcast in radio network models that assume a single channel and probabilistic message corruption. Finally, in recent work, Richa et al. [28] considered efficient MAC protocols in a single channel, multihop radio network, with an adversary that can cause a bounded amount of communication disruption.

2 Model

We model a synchronous multihop radio network with multiple communication channels, symmetric communication links, receiver collision detection, and adversarial disruption. In the following, for integer $x > 1$, let $[x] = \{1, ..., x\}$, and assume $\log$ denotes the base-2 logarithm. Fix an undirected graph $G = (V, E)$, with diameter D, where the vertexes in V correspond to the $n > 1$ processes in the network, which we uniquely label from $[n]$. We assume processes know n. To simplify notation we also assume that n is a power of 2. In this paper, when we denote a property holds *with high probability* (w.h.p.), we assume a probability of at least $1 - \frac{1}{n^x}$, for some sufficiently large positive integer x. Fix a set $[\mathcal{C}]$ of communication channels for some integer $\mathcal{C} \geq 1$, and a known upper bound on disruption, t, $0 \leq t \leq \mathcal{C}$. Executions in our model proceeds in synchronous rounds labeled $1, 2, \ldots$. Because we study broadcast problems, we assume processes can receive a message *from* and output a message *to* the environment, during each round. All processes start in round 1, but following the standard assumption made in the study of multihop broadcast (e.g., [3]), we assume no process can broadcast before it receives a message, either from another process or the environment.

To model disruption, we use the *t-disrupted model*, which was introduced in [13], and has since been extensively studied in the context of both single hop and multihop radio networks [11–14, 17, 26, 30, 31]. (See [11] for a good overview of this model and results.[7]) In the *t-disrupted model*, in each round r, an adversary chooses, for each process i, a set $disp(i, r)$ of up to t channels to *disrupt*. The adversary can use the history of the execution through round $r - 1$, as well as the process definitions, in deciding $disp(i, r)$. It does not, however, have advance knowledge of the random choices made in r. We consider two cases for the random choices: (i) *common randomness*, where processes can access a common source of random bits in each round, and (ii) *no common randomness*, case where the bits are independent at each process. Next, each process i chooses a channel $c \in [\mathcal{C}]$ on which to participate, and decides whether to *broadcast* or *receive*. If i broadcasts it receives nothing. If i receives, three behaviors are possible: (1) if no neighbor of i in G broadcasts on c in r and $c \notin disp(i, r)$, i detects silence, indicated by $\perp$; (2) if exactly one neighbor j of i in G broadcasts on c in r, and $c \notin disp(i, r)$, i receives j's message; (3) if two or more neighbors of i in G broadcast on c in r, or $c \in disp(i, r)$, i detects a collision, indicated by $\pm$. (That is, receiving on a disrupted channel is indistinguishable from detecting a collision.) Notice, i learns nothing about the activities of processes on other channels during this round.

The Multihop Broadcast Problem. Our goal in this paper is to define bounds for the *multihop broadcast problem*, which is defined as follows: At the beginning of round 1, a single *source* process is provided a message m by the environment. We say an algorithm *solves the multihop broadcast problem in r rounds* if and only if every process outputs m by round r, w.h.p.

The Local Broadcast Problem. In this paper, following the approach of [18], we decompose multihop broadcast into first solving local broadcast, and then using the construction presented in [18] to transform this local solution into a global one.

In more detail, the *T_A-local broadcast problem*, for positive integer T_A, assumes that the environment injects a message m at arbitrary processes at arbitrary times, and that every process that receives the message from the environment must eventually output *ack*. We say an algorithm *solves the T_A-local broadcast problem* if and only if the following hold: (a) If some process i receives the message from the environment in round r and outputs *ack* in round $r' \geq r$, then all neighbors of i output the message by round r', w.h.p. (b) We say a process is *active* in a given round r if it received the message from the environment in some round $r' \leq r$, and it has not yet output *ack* by the beginning of r. Given any interval of T_A rounds, if process i has a neighbor that is active in every round of the interval, then i outputs the message by the end of the interval, with *constant probability.*

Transforming Local Broadcast to Multihop Broadcast. The following theorem, which follows from Theorem 7.8 of [18], reduces the problem of multihop broadcast to local broadcast: [8]

[7] This model has been called many different names. Originally [13] it was unnamed; later works [11, 14] called it the *disrupted radio network* model; it was only in more recent work [26] that the more descriptive name of *t-disrupted* was introduced.

[8] Formally, the local broadcast problem described above is a simplified presentation of the *Abstract MAC Layer* formalism first introduced in [24]. The result cited from [18] provides an implementation of multihop broadcast that uses a probabilistic Abstract MAC Layer implementation. Our definition of local broadcast simplifies the Abstract MAC Layer definition down to only the properties needed to apply the transformation in [18]. In more detail, receiv-

Theorem 1 (Theorem 7.8 **of [18]).** *Given an algorithm that solves the T_A-local broadcast problem, we can construct an algorithm that solves the multihop broadcast problem in $O((D + \log n)T_A)$ rounds.*

3 Upper Bound for No Disruption

We begin with an algorithm for the case with no disruption (i.e., $t = 0$), that solves multihop broadcast in $O((D + \log n)(\log \mathcal{C} + \frac{\log n}{\mathcal{C}}))$ rounds. For $\mathcal{C} = 1$, this running time matches the canonical broadcast algorithm of Bar-Yehuda et al. [3], but as the number of channels increases so does our performance advantage. In Section 6, we will prove that for sufficiently large $\mathcal{C}$, this is within a $O(\log \log n)$ factor of optimal.

As described in Section 2, our approach is to first solve the *local* broadcast problem, then apply Theorem 1 to generate our *global* solution. Our algorithm only makes use of up to $\log n$ channels, so in this section we assume, w.l.o.g., $\mathcal{C} \leq \log n$.

Intuition. The key insight of our protocol is to trade channel diversity for time complexity. Most existing broadcast algorithms (e.g., [3]) described at a high level, have processes sequentially test $\log n$ different broadcast probabilities exponentially distributed between $1/n$ and $1/2$. For each process waiting to receive a message from transmitting neighbors, one of these probabilities should sufficiently reduce the contention and hence match what is needed to ensure that the message is delivered. Our algorithm, by contrast, leverages multiple channels to test multiple probabilities in parallel, gaining efficiency.

Our local broadcast algorithm consists of two subroutines: SEARCH and LISTEN. During, SEARCH, processes assign an exponential distribution of probabilities to the channels (captured by $schan$ in our algorithm description). A receiving process can then do a binary search over the channels (with silence indicating the probability is *too low*, and a collision indicating *too high*), to find the probability that best matches the number of transmitting neighbors. (This search is what necessitates receiver collision detection in our model.) If $\mathcal{C} \leq \log n$, however, then this SEARCH subroutine identifies only a rough range of $\log n/\mathcal{C}$ probabilities, in which is included the right probability for actually receiving a message. During the LISTEN subroutine, transmitting processes cycle through the different probabilities assigned to each channel (captured by $lchan$ in our algorithm description).[9] In both subroutines, care must be taken to account for the fact that many processes are both transmitters and receivers: a problem we solve by having processes choose a role with probability $1/2$.

Algorithm Description. The local broadcast algorithm has all processes alternate between executing the SEARCH and LISTEN subroutines presented in Figure 1, starting

ing a message m from the environment in our model corresponds to $bcast(m)$ in the Abstract MAC Layer, and outputting the message corresponds to calling $recv(m)$. In addition, the T_A parameter corresponds to $f_{prog}(\Delta)$, the constant probability of the T_A property holding corresponds to $1-\epsilon_{prog}$, and the high probability of all neighbors eventually outputting the message corresponds to $1 - \epsilon_{ack}$. We do not define an equivalent of f_{ack} or f_{rcv}, as neither are used in the transformation. We point the interested reader to [18] for more details.

[9] To make the probabilities work in our proofs, transmitters also try, for each channel, a constant number of probabilities from the neighboring channels. This is why $lchan$ cycles through $\log n/\mathcal{C} + O(1)$ different probability assignments, not just $\log n/\mathcal{C}$.

in round 1. Each call to SEARCH returns a candidate channel c_1, and the following call to LISTEN is made with $channel = c_1$. On receiving a message msg from the environment, a process sets $m \leftarrow msg$, and continues to try to transmit the message for the subsequent $AMAX$ calls to both subroutines, starting with the next call to SEARCH. After these $AMAX$ calls it outputs *ack*. In all other rounds, it sets $m \leftarrow \perp$. (Note, as required by our model, processes do not broadcast until they first receive a message.)

Constants Used in Algorithm. Let $k = \lceil \frac{\log n}{\mathcal{C}} \rceil$. The constant k represents the (approximate) number of probabilities assigned to each channel. Let $p_c = 1/2^{k(c-1)+1}$, for $c \in [\mathcal{C}]$. The function $schan()$ returns channel $c \in [\mathcal{C}]$ with probability p_c, and the *null* channel 0 with the sum of the remaining probability: $1 - \sum_{c \in [\mathcal{C}]} p_c$. That is, $schan$ chooses channels using an exponential probability distribution.

Next, we define a family of functions $lchan(r)$, for $r \in \{1, ..., k+3\}$. Intuitively, $lchan$ partitions the $\log n$ probabilities from $\{\frac{1}{n}, \frac{2}{n}, ..., \frac{1}{2}\}$ among the $\mathcal{C}$ channels. This means that k, defined above as $\lceil \frac{\log n}{\mathcal{C}} \rceil$, describes the number of probabilities in each channel partition. For each index passed to $lchan$, it assigns channels one of the probabilities from their partition, and then randomly selects a channel based on this distribution. The function $lchan(r)$ is defined as follows: if $r = 1$, it returns channel 1 with probability 0; if $r > k + 1$, it returns channel $\mathcal{C}$ with probability 0; if $r = k + 3$ and $k = 1$, it returns channel $\mathcal{C} - 1$ with probability 0; for all other r and c pairs, it returns channel c with probability $(2p_c)/2^{r-1}$. As with $schan()$, it returns the *null* channel 0 with the sum of the remaining probabilities for the given r value. The function $lchan$is defined for more than k values (i.e., $k + 3$ instead of k) because, to simplify the proof later, it helps if in addition to using every probability in a given channel's partition, we also use a constant number of probabilities that have been assigned to neighboring channels.

Finally, let $SMAX = 2(\lceil \log(\mathcal{C}) \rceil + 1)$, $LMAX = \lceil \frac{\log n}{\mathcal{C}} \rceil + 3$, and $AMAX = \Theta(\log n)$, where the constants are defined in our main theorem proof.

```
SEARCH($m$)
$c_1 \leftarrow 1$; $c_2 \leftarrow \mathcal{C}$; $count \leftarrow 1$
while $count \leq SMAX$
  $phase \leftarrow random(broadcast, listen)$
  if $(phase = broadcast)$ and $(m \neq \perp)$
    $b_c \leftarrow schan()$
    bcast($m, b_c$)
  else if $(phase = listen)$ and $(c_1 \neq c_2)$
    $channel \leftarrow \lceil c_1 + (c_2 - c_1)/2 \rceil$
    $rmsg \leftarrow$ recv($channel$)
    if $(rmsg = \perp)$ then $c_2 \leftarrow channel - 1$
    else $c_1 \leftarrow channel$
  $count \leftarrow count + 1$
```

```
LISTEN($m, channel$)
$count \leftarrow 1$
while $count \leq LMAX$
  $phase \leftarrow random(broadcast, listen)$
  if $(phase = broadcast)$ and $(m \neq \perp)$ then
    $b_c \leftarrow lchan(count)$
    bcast($m, b_c$)
  else
    $rmsg \leftarrow$ recv($channel$)
    if $(rmsg \neq \perp)$ and $(rmsg \neq \pm)$ then
      output($rmsg$)
  $count \leftarrow count + 1$
```

Fig. 1. The SEARCH and LISTEN subroutines called by our local broadcast solution. $SMAX = \Theta(\log \mathcal{C})$ and $LMAX = \Theta(\log n/\mathcal{C})$.

Correctness Proof. We prove that each pair of calls to SEARCH and LISTEN receives a message with constant probability, assuming there is a message to be received. We also prove that over AMAX calls to these subroutines, a message is received w.h.p. It follows that we solve T_A-local broadcast problem for $T_A = O(SMAX + LMAX)$, which when combined with Theorem 1 yields an algorithm that solves multihop broadcast problem in $O((D + \log n)(\log \mathcal{C} + \frac{\log n}{\mathcal{C}}))$ rounds.

To begin the proof, fix a process i and a call to SEARCH. Let I, $|I| \leq \Delta$, be the set of *active* neighbors of i during this call—that is, the neighbors of i with a message to send (i.e., $m \neq \perp$ in their call to SEARCH). We say a call to SEARCH is *valid* for this process i if and only if these following three conditions hold: (1) at the conclusion of the subroutine, $c_1 = c_2$; (2) for each **recv**(c), if $p_c|I| \leq \frac{1}{2}$, it returns $\perp$; and (3) for each **recv**(c), if $p_c|I| \geq 4$, it does not return $\perp$. (Otherwise, if a call to SEARCH is invalid, it may return a channel with too much or too little contention.) We prove this occurs with constant probability:

Lemma 1. *The call to SEARCH is valid with constant probability.*

Proof (Proof (sketch)). To prove the first condition of validity we must show that SEARCH sets $phase \leftarrow listen$ at least $\gamma \geq (\lceil \log(\mathcal{C}) \rceil + 1)$ times. Since this occurs according to a binomial distribution, with median $\frac{1}{2} \cdot SMAX \geq \gamma$, we conclude that with probability at least $\frac{1}{2}$ the SEARCH completes with $c_1 = c_2$.

To prove the second condition, let $\mathcal{L}$ contain every channel c such that i receives on c and assume $p_c|I| \leq \frac{1}{2}$. For a given $c \in \mathcal{L}$, the condition holds with probability $(1 - \frac{1}{2}p_c)^{|I|}$, and hence by a union bound, the condition holds over all relevant rounds with probability at least $1/2$. We prove the third condition in a similar manner, concluding that the condition holds over all relevant rounds with probability at least 0.8.

We now show that if process i's call to SEARCH is valid then, with constant probability, process i will receive a message during the subsequent call to LISTEN (assuming, of course, $|I| > 0$).

Lemma 2. *Suppose process i's call to SEARCH is valid and $|I| > 0$. Then, process i will receive a message during the subsequent LISTEN subroutine, with constant probability.*

Proof (Proof (sketch)). Let c be the channel returned by the call to SEARCH.

We consider two cases for the size of I. In the first case, assume $|I| = 1$. Here, $p_{c'}|I| \leq \frac{1}{2}$ for every channel $c' > 1$. Since we assume SEARCH was valid (with constant probability), every call to **recv** during the subroutine would return $\perp$. It follows that LISTEN executes on channel $c = 1$, where process i will receive a message with probability at least $\frac{1}{2} \cdot \frac{1}{2} \cdot p_1 = \frac{1}{8}$.

For the second case, assume $|I| > 1$. Let p_{min} be the smallest non-0 probability assigned to channel c in all $k + 3$ calls to $lchan$ in the listen phase, and let p_{max} be the largest probability. We can then bound both p_{max} and p_{min}: $p_{max}|I| \geq 1$ and $p_{min}|I| \leq 2$.

By definition, $p_{min} \leq p_{max}$. Combined, we conclude that there must exists a probability p', among those assigned to channel c by $lchan$ during LISTEN such that $1 \leq p'|I| \leq 2$. Consider the LISTEN round during which p' is assigned to channel c by $lchan$. During this round, process i will receive a message with probability $p_{rcv} \geq \frac{1}{2}p''|I|(1 - p'')^{|I|-1}$, where $p'' = \frac{p'}{2}$ is the probability that a process broadcasts in channel c in that round, and the first $\frac{1}{2}$ bounds the probability that i receives. Note that $\frac{1}{2} \leq p''|I| \leq 1$ and $p'' \leq \frac{1}{2}$. We now simplify p_{rcv}: $p_{rcv} = \frac{1}{2}p''|I|\,(1-p'')^{|I|-1} \geq \frac{1}{2}p''|I|\,(1-p'')^{|I|}$. This later term is greater than or equal to $\frac{1}{2}p''|I|\left(\frac{1}{4}\right)^{p''|I|} \geq \frac{1}{2} \cdot \frac{1}{4} = \frac{1}{8}$. Notice, the third step uses our above-stated fact that $p'' \leq \frac{1}{2}$, and the fourth step uses the other above-stated fact that $\frac{1}{2} \leq p''|I| \leq 1$.

We can now prove that the algorithm solves the local broadcast problem.

Lemma 3. *The algorithm solves the* $2(SMAX + LMAX)$*-local broadcast problem.*

Proof. By Lemmas 1 and 2, we know the algorithm satisfies property (b) of the local broadcast problem, for $T_A = 2(SMAX + LMAX)$ (the factor of 2 accounts for the case that a message arrives after SEARCH has begun, necessitating we wait until the next call to SEARCH begins before the process begins trying to send the message). To show the algorithm satisfies property (a), assume that some process i receives the message from the environment for the first time at some round r. Let j be a neighbor of i. By our above argument, over the next $AMAX$ pairs of calls to SEARCH and LISTEN, j will receive the message from i (or another neighboring process) with some constant probability p. Process j therefore fails to receive the message in all $AMAX$ pair of calls, with probability no greater than $(1-p)^{AMAX} \leq e^{-p \cdot AMAX}$. Because p is constant and $AMAX = O(\log n)$, for sufficiently large constant factors, this failure with probability no more than $\frac{1}{n^{x+1}}$, for any positive constant x. By a union bound over the $O(n)$ neighbors of i, property (a) holds w.h.p., as needed.

Given Lemma 3, we can now apply Theorem 1 to derive our final result:

Theorem 2. *We can construct an algorithm that solves the multihop broadcast problem with no disruption (*$t = 0$*) in* $O((D + \log n)(\log \mathcal{C} + \frac{\log n}{\mathcal{C}}))$ *rounds.*

4 Upper Bound for Disruption and Common Randomness

In this section, we assume that channels may be disrupted (i.e., $t > 0$) and that processes have access to a common source of randomness. We present an algorithm that solves the multihop broadcast problem in $O((D + \log n)(\frac{\mathcal{C} \log \mathcal{C} \log\log \mathcal{C}}{\mathcal{C}-t} + \frac{\log n}{\mathcal{C}-t}))$ rounds, where t is the known upper bound on disrupted channels. Therefore, for even large amounts of disruption (i.e., for any t up to a constant factor of $\mathcal{C}$) our disruption-tolerant protocol performs only a factor of $O(\log\log \mathcal{C})$ slower than our no disruption protocol from Section 3. This means that for sufficiently large $\mathcal{C}$, we still outperform the best possible single channel solution in many cases, and are more efficient than the canonical single channel algorithm of [3]. It follows that common randomness is a potent weapon against disruptive interference.

Intuition. Our approach is to extend the no disruption algorithm from Section 3. In more detail, we replace the broadcast and received primitives of the no disruption protocol with disruption-tolerant versions that use coordinated frequency hopping (specified by the common random bits) to evade disruption. We show that the new **sim-recv** subroutine outputs the same value as its no disruption counterpart (i.e., the **recv** subroutine) with *just enough* probability to ensure that our analysis still applies. Note that the

```
**sim-bcast**$_\gamma(m, c)$:
$simcount \leftarrow 1$
while $simcount \leq \gamma$
    $\psi \leftarrow$ *a channel permutation generated with common source of randomness for this round.*
    **bcast**$(m, \psi(c))$
    $simcount \leftarrow simcount + 1$
```

```
**sim-recv**$_\gamma(c)$
$rmsg \leftarrow \pm$; $simcount \leftarrow 1$
while $simcount \leq \gamma$
    $\psi \leftarrow$ *a channel permutation generated with common source of randomness for this round.*
    $m \leftarrow$ **recv**$(\psi(c))$
    if $m \neq \pm$ then $rmsg \leftarrow m$
    $simcount \leftarrow simcount + 1$
return $rmsg$
```

Fig. 2. The simulated broadcast and receive functions that replace the **bcast** and **recv** functions of the no disruption algorithm (Figure 1) to produce a local broadcast algorithm for the setting with disruption and a common source of randomness. For SEARCH, $\gamma = \Theta(\frac{\mathcal{C}}{\mathcal{C}-t} \log\log \mathcal{C})$, and for LISTEN, $\gamma = \Theta(\frac{\mathcal{C}}{\mathcal{C}-t})$.

new subroutines call the **bcast** and **recv** functions several times, but not so much that the running time becomes unwieldy.

Algorithm Description. Our local broadcast algorithm replaces each call to **bcast** and **recv** in the no disruption subroutines from Figure 1, with calls to *simulated* broadcasts and receives that use multiple rounds to evade disruption. In more detail, in our modified version of the SEARCH subroutine from Figure 1, which we call DSEARCH, we replace each call to **bcast**(m, b_c) with a call to **sim-bcast**$_{\gamma_S}(m, b_c)$, and each call to **recv**$(channel)$ with a call to **sim-recv**$_{\gamma_S}(channel)$, where **sim-bcast** and **sim-recv** are defined in Figure 2, and $\gamma_S = \Theta(\frac{\mathcal{C}}{\mathcal{C}-t} \log\log \mathcal{C})$. For the modified LISTEN subroutine, which we call DLISTEN, we do the same replacement of **bcast** and **recv** with **sim-bcast** and **sim-recv**, substituting $\gamma_L = \Theta(\frac{\mathcal{C}}{\mathcal{C}-t})$ for γ_S. For any round r of an execution, we assume that every process generating the random channel permutation ψ during r, will generate the same permutation, using the common randomness.

Correctness Proof. We begin by bounding the probability that our simulated bcast and receives behave the same as if we were in the no disruption setting. This claim follows primarily from the fact that the probability that the adversary disrupts every channel in $\psi(c)$ in the relevant round is $O(1/\log \mathcal{C})$.

Lemma 4. *Suppose process i calls* **sim-recv** *during some round of DSEARCH, and all of i's neighbors also call either* **sim-bcast** *or* **sim-recv** *during this same round. With probability at least $1 - O(\frac{1}{\log \mathcal{C}})$,* **sim-recv** *will return i the same value as if these same processes had called* **bcast** *and* **recv***, with the same parameters, in the setting where $t = 0$.*

If we replace γ_S with γ_L, we can show a similar result for DLISTEN, this time with constant probability:

Lemma 5. *Suppose process i calls* **sim-recv** *during some round of DSEARCH, and all of i's neighbors also call either* **sim-bcast** *or* **sim-recv** *during this same round. With constant probability,* **sim-recv** *will return i the same value as if these same processes had called* **bcast** *and* **recv***, with the same parameters, in the setting where $t = 0$.*

We now show, much as in Section 3, that the local broadcast performs well:

Lemma 6. *The algorithm solves the* $2(SMAX \cdot \gamma_S + LMAX \cdot \gamma_L)$*-local broadcast problem.*

Given Lemma 6, we apply Theorem 1 to derive our final result regarding multihop broadcast:

Theorem 3. *We can construct an algorithm that solves the multihop broadcast problem with common randomness in* $O((D + \log n)(\frac{\mathcal{C} \log \mathcal{C} \log \log \mathcal{C}}{\mathcal{C}-t} + \frac{\log n}{\mathcal{C}-t}))$ *rounds.*

5 Upper Bound for Disruption and No Common Randomness

In this section, we assume disruption and no common source of randomness. We present an algorithm that solves multihop broadcast in $O((D + \log n)\frac{\mathcal{C}t}{\mathcal{C}-t} \log(\frac{n}{t}))$ rounds. In Section 6, we prove this to be within a factor of $O(\log(\frac{n}{t}))$ of optimal. Unlike the common randomness case, here we actually perform (slightly) worse than the single channel algorithm of [3] (at least, for large t). This difference, however, is bounded by a factor of $O(\log n)$, which is arguably still a reasonable price to pay for the increased robustness. In the following, we assume w.l.o.g. that $\mathcal{C} < 2t$.

Intuition. There are three basic challenges to overcome: First, because some t channels are disrupted, processes must attempt to communicate on more than t channels, and to avoid the disruption, the communication must be randomized. Second, since the processes have no source of common randomness, the random channel selection potentially delays the receivers from finding the broadcasts. Third, processes still have to solve the problem of contention, i.e., the fact that many broadcasters may be competing to send a message. To overcome these problems, we have processes repeatedly choose channels uniformly at random, cycling through the $\log n$ broadcast probabilities that are exponentially distributed between $1/n$ and $1/2$.

Algorithm Description. Our local broadcast algorithm works as follows. First, the execution is divided into epochs of length $\lceil \log n/\mathcal{C} \rceil$. If a message is injected at a process v in some round r, then process v waits until the beginning of the next epoch before trying to disseminate the message. We say that a process that has received message m by the first round of some epoch e, but has not yet returned an acknowledgment for m, *participates* in epoch e. In each round of an epoch, each participating process decides whether to broadcast and on which channel. In particular, in round r, a participating process v broadcasts with probability $1/2^r$; it chooses channel $c \in [\mathcal{C}]$ with probability $1/\mathcal{C}$. Every process u that is not broadcasting a message chooses a channel on which to listen with the same uniform probability $1/\mathcal{C}$. A process v returns an acknowledgment when it has participated for $\Theta((\mathcal{C}^2 \log n)/(\mathcal{C} - t))$ epochs.

Correctness. We argue that this protocol solves T_A-local broadcast for $T_A = O((\mathcal{C}^2 \log n/C)/(\mathcal{C} - t))$. We first argue that if process u has a participating neighbor in epoch e, then by the end of the epoch, it receives the message with constant probability:

Lemma 7. *Let* u *be a process that has not received the message* m *prior to epoch* e*. Let* V *be the set of neighbors of* u *participating in epoch* e*, and assume that* $|V| > 0$*. Then with probability at least* $(\mathcal{C} - t)/(32\mathcal{C}^2)$*,* u *receives message* m *by the end of epoch* e*.*

Proof (Proof (sketch)). Process u receives the message m in a round r if the following three conditions are satisfied: (a) there is exactly one process in $v \in V$ that broadcasts in round r; (b) the channel selected by v is not disrupted in round r; and (c) process u chooses to listen on the same channel on which v broadcasts in round r. Consider round $r = \lceil \log |V| \rceil$ in epoch e. We now bound the probability that these three events occur.

Let $c \in \mathcal{C}$ be the channel chosen by u in round r of epoch e. With probability $(\mathcal{C} - t)/\mathcal{C}$ we observe that c is not disrupted. We calculate the probability that exactly one participating process in V broadcasts on channel c: $\sum_{v \in V} \frac{1}{\mathcal{C}2^r} \left(1 - \frac{1}{\mathcal{C}2^r}\right)^{|V|-1} \geq 1/(32\mathcal{C})$. Thus, with probability $(\mathcal{C} - t)/(32\mathcal{C}^2)$, process u receives the message by the end of the epoch.

From this we conclude that the protocol solves the T_A-local broadcast problem:

Lemma 8. *The specified protocol solves the T_A-local broadcast problem for $T_A = O(\frac{\mathcal{C}^2 \log(n/\mathcal{C})}{(\mathcal{C}-t)})$.*

Proof. First, we argue that every process with active neighbors receives the message within time T_A with constant probability. Consider a process u. By Lemma 7, during each epoch in which a neighbor participates, u receives the message with probability $(\mathcal{C} - t)/(32\mathcal{C}^2)$. Thus, over $(32\mathcal{C}^2/(\mathcal{C} - t))$ epochs, process u receives the message with constant probability. An active neighbor may not participate for the first epoch when it receives the message, and from this we conclude that if $T_A = [(32\mathcal{C}^2/(\mathcal{C} - t)) + 1] \log(n/\mathcal{C})$, then u receives the message as required.

Next, we argue that when a node sends an acknowledgment, every neighboring process has received the message. Specifically: in each epoch, each neighbor receives the message with constant probability. Thus within $O(\log n)$ epochs, every neighbor has received the message with high probability, as required.

Since $t < \mathcal{C} \leq 2t$, we can apply Theorem 1 to conclude:

Theorem 4. *We can construct an algorithm that solves the multihop broadcast problem without common randomness in $O((D + \log n)(\frac{\mathcal{C}t}{\mathcal{C}-t}) \log(\frac{n}{t}))$ rounds.*

6 Lower Bounds

We begin by showing that the $O((D + \log n)(\log \mathcal{C} + \frac{\log n}{\mathcal{C}}))$-time broadcast algorithm from Section 3 is (almost) tight for sufficiently large $\mathcal{C}$, by proving a $\Omega(D + \frac{\log^2 n}{\mathcal{C}})$ lower bound for solving broadcast in this setting. (In more detail, for $\mathcal{C} = \Omega(\log n)$, the upper bound is within a factor of $O(\log \log n)$ of the lower.)

Theorem 5. *For any $D \leq n/2$: every multihop broadcast algorithm requires $\Omega(D + \frac{\log^2 n}{\mathcal{C}})$ rounds.*

Proof. We first note that we can simulate any protocol for a network with $\mathcal{C} > 1$ in a network where $\mathcal{C} = 1$. In more detail, we use $\mathcal{C}$ rounds in the single channel network to simulate each round from the multi-channel network, with each simulation round being dedicated to a different channel. It follows that if $f(n)$ is a lower bound for multihop broadcast in a network where $\mathcal{C} = 1$, then $f(n)/\mathcal{C}$ is a lower bound for networks with larger $\mathcal{C}$. The question remains what lower bounds apply to our network model with

$\mathcal{C} = 1$. The commonly cited $\Omega(D \log (n/D))$ bound of Kushilevitz and Mansour [25], proved for randomized distributed multihop broadcast, does *not* apply in our setting, as we assume receiver collision detection. In fact, there are no bounds, that we know of, specific to distributed broadcast with collision detection. With this in mind, we turn to the bound for *centralized* solutions to broadcast in single channel networks, from [2]. This bound proves that there exists a family of constant-diameter graphs such that every centralized broadcast algorithm requires at least $f(n) = \Omega(\log^2 n)$ rounds. Centralized solutions, of course, are stronger than randomized distributed solutions with collision detection, so a bound for the former certainly holds for the latter. By our above simulation argument, it holds that no algorithm can solve multihop broadcast in less than $f(n)/\mathcal{C} = \Omega(\frac{\log^2 n}{\mathcal{C}})$ rounds. If we replace n with $n - D$, due to our assumption that $D \leq n/2$ we get a network of size $O(n)$ that still requires $\Omega(\frac{\log^2 n}{\mathcal{C}})$ rounds to broadcast in. If we put this network on one end of a line of D nodes, and make the far end the broadcast source, the bound extends to $\Omega(D + \frac{\log^2 n}{\mathcal{C}})$.

We now continue with a lower bound for the setting with disruption ($t > 0$) and *no* common source of randomness. In Section 5, we presented a $O((D+\log n)\frac{\mathcal{C}t}{\mathcal{C}-t}\log(\frac{n}{t}))$-time broadcast algorithm in this setting. Our lower bound below shows this to be within a factor of $O(\log(\frac{n}{t}))$ of optimal. This bound uses the following fact, first proved in our study of the wireless synchronization problem in the t-disrupted model [12]:

Lemma 9 (Theorem 4 of [12]). *Assume there are two processes u and v attempting to communicate in a t-disrupted network with $\mathcal{C}$ channels, $t > 0$, and no common source of randomness. Fix a constant ϵ. With probability ϵ, u and v cannot communicate for $\Omega(\frac{\mathcal{C}t}{\mathcal{C}-t}\log(1/\epsilon))$ rounds.*

We use this fact to prove our bound on broadcast:

Theorem 6. *Assume no common source of randomness. It follows that every algorithm requires $\Omega((D + \log n)\frac{\mathcal{C}t}{\mathcal{C}-t})$ rounds to solve the multihop broadcast problem.*

Proof. We consider two different networks. First, consider the simple network with only two processes, u and v. Lemma 9 shows that for $\epsilon = 2/n$ there is a probability of at least $2/n$ that u and v do not communicate for $\Omega(\log n \frac{\mathcal{C}t}{\mathcal{C}-t})$ rounds.

Next, consider the "line" network consisting of a set of processes $v_0, v_2, ..., v_D$, where v_0 is the source and can communicate only with v_1, and, for $0 < i < D$, v_i can communicate only with v_{i-1} and v_{i+1}. Fix $\epsilon' = 1/4e$. By Lemma 9, we know that with probability $1 - \epsilon'$, for some constant c, it takes v_i at least $c(\frac{\mathcal{C}t}{\mathcal{C}-t})$ rounds to transmit the message to v_{i+1}.

We now calculate the probability that for some $D/2$ of the v_i, the communication from v_i to v_{i+1} is faster than $c(\frac{\mathcal{C}t}{\mathcal{C}-t})$. In particular, for a given set of $D/2$ links, the probability is $\epsilon'^{D/2}$ that each communication from v_i to v_{i+1} is faster than $c(\frac{\mathcal{C}t}{\mathcal{C}-t})$. Moreover, there are at most $\binom{D}{D/2} \leq (2e)^{D/2}$ such sets of $D/2$ links. Thus, for $\epsilon' < 1/4e$, we conclude that the probability of $D/2$ links exceeding the specified speed is at most $(2e\epsilon')^{D/2} < (1/2)^{D/2} \leq 1/2$ (where $D > 1$). Thus, with probability at least $1/2$, half the links require time $\Omega(\frac{\mathcal{C}t}{\mathcal{C}-t})$, leading to a running time of $\Omega(D\frac{\mathcal{C}t}{\mathcal{C}-t})$. Combining these two claims yields the desired result.

7 Conclusion and Future Work

In this paper, we study the problem of multihop broadcast in a radio network model that assumes multiple channels and a bounded amount of adversarial disruption. We show that additional communication channels can add both *efficiency* (as compared to the single channel setting) and *robustness* (in terms of resilience to a bounded amount of adversarial communication disruption). These advantages are especially pronounced if we assume a common source of randomness. This reinforces our belief that broadcast algorithms should better leverage the multiple communication channels made available today by most popular radio protocols.
An interesting future work is to relax the assumption on the knowledge of an upper bound on t, and design algorithms that perform with running time relative to the actual amount of adversarial disruption. Another interesting future work is to design deterministic solutions that leverage the multi-selectors introduced in [17].

References

1. IEEE 802.11. Wireless LAN MAC and Physical Layer Specifications, June 1999.
2. N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. A Lower Bound for Radio Broadcast. *Journal of Computer System Sciences*, 43(2):290–298, 1991.
3. 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.
4. Bluetooth Consortium. *Bluetooth Specification Version 2.1*, July 2007.
5. I. Chlamtac and S. Kutten. On Broadcasting in Radio Networks: Problem Analysis and Protocol Design. *IEEE Transactions on Communications*, 33(12):1240–1246, 1985.
6. I. Chlamtac and O. Weinstein. The wave expansion approach to braodcasting in multihop radio networks. *IEEE Transactions on Communications*, 39:426–433, 1991.
7. 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.
8. Bogdan S. Chlebus, Leszek Gasieniec, Alan Gibbons, Andrzej Pelc, and Wojciech Rytter. Deterministic Broadcasting in Unknown Radio Networks. In *Proceedings of the Symposium on Discrete Algorithms*, 2000.
9. A. Clementi, A. Monti, and R. Silvestri. Round Robin is Optimal for Fault-Tolerant Broadcasting on Wireless Networks. *Journal of Parallel and Distributed Computing*, 64(1):89–96, 2004.
10. Artur Czumaj and Wojciech Rytter. Broadcasting Algorithms in Radio Networks with Unknown Topology. In *Proceedings of the Symposium on Foundations of Computer Science*, 2003.
11. Shlomi Dolev, Seth Gilbert, Rachid Guerraoui, Dariusz R. Kowalski, Calvin Newport, Fabian Kohn, and Nancy Lynch. Reliable Distributed Computing on Unreliable Radio Channels. In *the Proceedings of the 2009 MobiHoc S^3 Workshop*, 2009.
12. Shlomi Dolev, Seth Gilbert, Rachid Guerraoui, Fabian Kuhn, and Calvin Newport. The Wireless Synchronization Problem. In *Proceedings of the International Symposium on Principles of Distributed Computing*, 2009.
13. Shlomi Dolev, Seth Gilbert, Rachid Guerraoui, and Calvin Newport. Gossiping in a Multi-Channel Radio Network: An Oblivious Approach to Coping with Malicious Interference. In *Proceedings of the International Symposium on Distributed Computing*, 2007.

14. Shlomi Dolev, Seth Gilbert, Rachid Guerraoui, and Calvin Newport. Secure Communication Over Radio Channels. In *Proceedings of the International Symposium on Principles of Distributed Computing*, 2008.
15. Vadim Drabkin, Roy Friedman, and Marc Segal. Efficient byzantine broadcast in wireless ad hoc networks. In *Proceedings of the Conference on Dependable Systems and Networks*, pages 160–169, 2005.
16. L. Gasieniec, D. Peleg, and Q. Xin. Faster Communication in Known Topology Radio Networks. *Distributed Computing*, 19(4):289–300, 2007.
17. Seth Gilbert, Rachid Guerraoui, Darek Kowalski, and Calvin Newport. Interference-Resilient Information Exchange. In *the Proceedings of the Conference on Computer Communication*, 2009.
18. Majid Khabbazian, Dariusz Kowalski, Fabian Kuhn, and Nancy Lynch. Decomposing Broadcast Algorithms Using Abstract MAC Layers. In *Proceedings of the International Workshop on Foundations of Mobile Computing*, 2010.
19. C-Y. Koo. Broadcast in radio networks tolerating byzantine adversarial behavior. In *Proceedings of the International Symposium on Principles of Distributed Computing*, pages 275–282, 2004.
20. Chiu-Yuen Koo, Vartika Bhandari, Jonathan Katz, and Nitin H. Vaidya. Reliable broadcast in radio networks: The bounded collision case. In *Proceedings of the International Symposium on Principles of Distributed Computing*, 2006.
21. D. Kowalski and A. Pelc. Broadcasting in Undirected Ad Hoc Radio Networks. In *Proceedings of the International Symposium on Principles of Distributed Computing*, 2003.
22. D. Kowalski and A. Pelc. Time of Deterministic Broadcasting in Radio Networks with Local Knowledge. *SIAM Journal on Computing*, 33(4):870–891, 2004.
23. D. Kowalski and A. Pelc. Optimal Deterministic Broadcasting in Known Topology Radio Networks. *Distributed Computing*, 19(3):185–195, 2007.
24. Fabian Kuhn, Nancy Lynch, and Calvin Newport. The Abstract MAC Layer. In *Proceedings of the International Symposium on Distributed Computing*, 2009.
25. E. Kushilevitz and Y. Mansour. An $\Omega(D \log(N/D))$ Lower Bound for Broadcast in Radio Networks. *SIAM Journal on Computing*, 27(3):702–712, 1998.
26. Calvin Newport. *Distributed Computation on Unreliable Radio Channels*. PhD thesis, MIT, 2009.
27. Andrzej Pelc and David Peleg. Feasibility and Complexity of Broadcasting with Random Transmission Failures. In *Proceedings of the International Symposium on Principles of Distributed Computing*, 2005.
28. A. Richa, C. Scheideler, S. Schmid, and J. Zhang. A Jamming-Resistant MAC Protocol for Multi-Hop Wireless Networks. In *Proceedings of the International Symposium on Distributed Computing*, 2010.
29. Johannes Schneider and Roger Wattenhofer. What Is the Use of Collision Detection (in Wireless Networks)? In *Proceedings of the International Symposium on Distributed Computing*, 2010.
30. Mario Strasser, Christina Pöpper, and Srdjan Capkun. Efficient Uncoordinated FHSS Anti-jamming Communication. In *Proceedings International Symposium on Mobile Ad Hoc Networking and Computing*, 2009.
31. Mario Strasser, Christina Pöpper, Srdjan Capkun, and Mario Cagalj. Jamming-resistant Key Establishment using Uncoordinated Frequency Hopping. In *the Proceedings of the IEEE Symposium on Security and Privacy*, 2008.