Optimal Broadcast in Shared Spectrum Radio Networks*

Mohsen Ghaffari[1], Seth Gilbert[2], Calvin Newport[3], and Henry Tan[3]

[1] MIT
ghaffari@mit.edu
[2] National University of Singapore
seth.gilbert@comp.nus.edu.sg
[3] Georgetown University
{cnewport,ztan}@cs.georgetown.edu

Abstract. This paper studies single hop broadcast in a single hop shared spectrum radio network. The problem requires a source to deliver a message to n receivers, where only a polynomial upper bound on n is known. The model assumes that in each round, each device can participate on 1 out of $\mathcal{C} \geq 1$ available communication channels, up to $t < \mathcal{C}$ of which might be disrupted, preventing communication. This disruption captures the unpredictable message loss that plagues real shared spectrum networks. The best existing solution to the problem, which comes from the systems literature, requires $O\big(\frac{\mathcal{C}t}{\mathcal{C}-t} \log n\big)$ rounds. Our algorithm, by contrast, solves the problem in $O\big(\frac{\mathcal{C}}{\mathcal{C}-t} \lceil \frac{t}{n} \rceil \log n\big)$ rounds, when $\mathcal{C} \geq \log n$, and in $O\big(\frac{\mathcal{C}}{\mathcal{C}-t} \log n \cdot \log\log n\big)$ rounds, when $\mathcal{C}$ is smaller. It accomplishes this improvement by deploying a self-regulating relay strategy in which receivers that already know useful information coordinate themselves to efficiently assist the source's broadcast. We conclude by proving these bounds tight for most cases.

1 Introduction

Consider a wireless device, which we call the *source*, with a message to send to a group of nearby devices, which we call the *receivers*. If this network had a dedicated communication channel, the problem would be easily solved: the receivers could simply wait on this channel for the source to broadcast its message. Unfortunately, in practice, this assumption almost never holds. Most wireless networking now takes place in *shared spectrum networks* where a group of communication frequencies are shared, in an uncoordinated manner, by multiple different networks, protocols, and unrelated sources of interference. The 2.4 GHz band, for example, is used by 802.11, Bluetooth, Zigbee, many different types of sensor network motes, cordless phones, baby monitors, and some types of car alarm sensors. Not surprisingly, interference between these competing devices is common [8].

The challenge faced by our source is now more pronounced. It can no longer use a fixed channel to communicate, because that channel might be disrupted by other users of the same spectrum. It must instead start sifting through the channels, seeking its receivers amidst this churning sea of electromagnetic noise. This problem, which we

* This research was supported by Singapore NUS FRC grant R-252-000-443-133 and the Ford Motor Company University Research Program.

call *Single-hop Shared-spectrum Broadcast* (SSB), comes from the systems literature, where it is well-studied [9–12, 16–21]. In practice, SSB algorithms are typically used to send a session key from a master device to its slaves. This key can then be used to configure more traditional disruption-resilient coding techniques such as frequency-hopping spread spectrum (FHSS) or direct sequence spread spectrum (DSSS). A well-known example of an SSB solution is the pairing protocol used by Bluetooth. (See [17] for more on the practical motivation driving this problem.)

Results. Following the lead of past theory work on shared spectrum, we formally describe this setting using the *t-disrupted* network model [2–7, 13, 14], which assumes devices have access to $\mathcal{C} \geq 1$ communication channels, with up to $t < \mathcal{C}$ disrupted by outside sources of interference. In this model, in each round, each device chooses a single channel on which to participate. The set of up to t disrupted channels is chosen arbitrarily and can change from round to round.

The best existing SSB algorithm [20], when analyzed in this model, solves the problem in $O\big(\frac{\mathcal{C}t}{\mathcal{C}-t}\log n\big)$ rounds, with high probability (i.e., at least $1 - 1/N$, where N is a known polynomial upper bound on n). In Section 3, we describe *pandemic broadcast*, a pair of algorithms which solve the problem in $O\big(\frac{\mathcal{C}}{\mathcal{C}-t}\lceil\frac{t}{n}\rceil\log n\big)$ rounds, for $\mathcal{C} \geq \log n$, and in $O\big(\frac{\mathcal{C}}{\mathcal{C}-t}\log n \cdot \log\log n\big)$ rounds for smaller $\mathcal{C}$. In the $\mathcal{C} \geq \log n$ setting, our solution is a factor of t faster than the existing solution when $t \leq n$, and a factor of n faster when $t > n$. For the small $\mathcal{C}$ setting, this advantage reduces only slightly to $t/\log\log n$. Finally, in Section 4, we prove our solutions optimal (within $\log\log n$ factors) for most relevant cases.

Because the SSB problem is drawn directly from the systems literature, our solutions can be applied directly back to these systems, making it a good example of distributed algorithm theory helping to improve real world wireless networks.

Intuition. The core insight driving our algorithm is that it exploits the parallelism inherent in having multiple available communication channels. The best existing solution [20] allows devices to only receive information from the source. In contrast, our solutions, which we call *Pandemic Broadcast Algorithms*, allow devices to relay information on behalf of the source, eventually converging on a state where there is a single relayer per channel, on a constant fraction of the channels. In this state, the remaining uninformed devices quickly learn new information.

The main challenge in implementing this idea is the unknown number of receivers. If too few devices relay, then not much advantage is gained. On the other hand, if too many devices relay, they clog the channels with collisions. The pandemic broadcast algorithms overcome these issues with a self-regulating process, inspired by infectious disease propagation. In more detail, the algorithms rely on two-stage infection. The first stage aggressively infects receivers, turning them into relayers. It guarantees at least $\mathcal{C}$ relayers, but may end up producing many more before it dies out. The second stage has these relayers reduce their numbers back down to around $\mathcal{C}$, allowing them to efficiently infect the remaining receivers. This reduction relies on a pair of distributed estimation subroutines, interesting in their own right. The first routine, which requires at least $\log n$ channels, gains efficiency by moving estimation from the time domain to the channel domain. In more detail, it uses $\log n$ channels to estimate the number of relayers in a single round instead of using $\log n$ rounds to estimate the relayers using a single channel. The second routine, which works for small number of channels (and is slightly slower), uses the source to emulate collision detection, allowing the relayers to run an efficient estimation routine on a single (changing) channel.

Related Work. The SSB problem was introduced in [19], which described an algorithm that delivers $k \geq 1$ messages in $O\big(\frac{\mathcal{C}^2}{\mathcal{C}-t}k\log n\big)$ rounds.[1] In [18, 20], erasure coding on the packets, and more clever use of the channels when t was small, improved the result to $O\big(\frac{\mathcal{C}t}{\mathcal{C}-t}(\log n + k)\big)$ rounds. In this paper, we focus on the case where $k = 1$, yielding $O\big(\frac{\mathcal{C}t}{\mathcal{C}-t}\log n\big)$ as the most relevant comparable result. It was recently suggested in [21] that relaying could be used to speed up SSB. However, the algorithms presented in [21] make strong assumptions. The algorithm described here uses the general idea of relaying from [21], but differs in essentially all other details. Concurrent to the series of systems papers cited above, another series looked at solving SSB using uncoordinated spread spectrum coding techniques [9–12, 16]. This approach is less immediately applicable as it requires modifications to the radio.

On the theory side, the shared spectrum model we use to study the SSB problem, sometimes called the *t-disrupted model*, has been previously used to study all-to-all gossip [5–7], pairwise node discovery [13], leader election [2, 3], and multihop broadcast (with collision detection) [4]. In [14], it was shown how to simulate a reliable channel in this shared spectrum setting, simplifying the development and analysis of reliable algorithms (though often at the cost of added time complexity). We underscore that the t-disrupted model does a good job of describing real shared spectrum networks by noting that it is cited in some of the systems SSB papers [19, 20].

The broadcast problem is well-studied in the the classical wireless model where devices shares a single dedicated undisrupted channel. The seminal result of Bar-Yehuda et al. [1], for example, solves broadcast in $O((D + \log n)\log n)$ for a multihop network of D hops, which is (near) optimal. The single hop broadcast problem, however, is trivially solved in this undisrupted setting, as the distinguished source can broadcast without disruption or contention. In our shared spectrum model, the broadcast algorithm of [4] can be adapted to solve our problem in $O\big(\frac{\mathcal{C}}{\mathcal{C}-t}t\log n\log(n/t)\big)$ rounds, assuming collision detectors and sufficiently large $\mathcal{C}$. This solution, even though it uses collision detectors (which our algorithms do not), is still a factor of t slower than ours under comparable conditions.

2 Model and Problem

We model a single hop synchronous wireless network consisting of $\mathcal{C} \geq 1$ communication channels and $n \geq 2$ devices. We assume each device runs the same randomized *algorithm*, and we refer to each individual device executing this algorithm as a *process*. All processes start an execution together in round 1. In each round, each process i chooses a single channel $c \in \{1, ..., \mathcal{C}\}$ on which to participate. Concurrently, an abstract *interference adversary* chooses up to t, $1 \leq t < \mathcal{C}$, channels to *disrupt*.

[1] The existing SSB papers cited here are from the systems literature and therefore do not analyze the time complexity of their algorithms asymptotically, in the way that is standard for the theory literature. We calculated the time complexities shown here based on how their algorithms would perform in our formal model with our parameters.

We model this adversary as an arbitrary randomized algorithm that receives no inputs during an execution. Therefore, its disruption strategy *can* be based on the algorithm executed by the processes (if, for example, processes hard-code a frequency hopping pattern in their definition, the adversary can disrupt that pattern). On the other hand, it *cannot* base its disruption on the random bits used by processes during the execution or the content of messages sent. This captures the reality of disruption in shared spectrum networks which tends to fall somewhere between random and malicious.[2]

A process i participating on channel c during round r receives a message m if and only if: (1) c is not disrupted in r; and (2) only one process broadcasts on c during r, and it broadcasts m (i.e., concurrent broadcasts on a channel leads to collision). To make our upper bound as strong as possible, we assume processes cannot distinguish collisions, disruption, and silence. We assume that processes know t and a polynomial upper bound on n, denoted N, but not n itself.

Formally, the SSB problem assumes a single *source* with a message to send to the remaining processes, which we call *receivers*. We say an algorithm solves the SSB problem in $f(n, \mathcal{C}, t)$ rounds if it guarantees that the source delivers the message to all receivers in $f(n, \mathcal{C}, t)$ rounds, with high probability, i.e., probability at least $1 - 1/n$.

3 Upper Bounds

In this section, we describe and analyze a pair of SSB algorithms, called *pandemic broadcast algorithm 1* (PBA1) and *pandemic broadcast algorithm 2* (PBA2). We use PBA1 when $\mathcal{C} \geq \log N$ and PBA2 when $\mathcal{C} < \log N$, where N is the aforementioned polynomial upper bound on n. In most real shared spectrum networks, $\mathcal{C}$ will be typically larger than $\log N$, as such PBA2 is presented mainly for completeness.[3] We begin, in Section 3.1, by describing these algorithms for the case where $t \leq 0.05 \times \mathcal{C}$, which we call the *low-disruption* regime. Later, in Section 3.2, we show how to simulate these protocols, with an overhead factor of $\frac{\mathcal{C}}{\mathcal{C}-t}$, for the case where $t > 0.05 \times \mathcal{C}$, which we call the *high-disruption* regime.

3.1 Low-Disruption Regime

We begin by studying the case where no more than a constant fraction of the channels can be disrupted concurrently. That is, in this section we assume $t \leq 0.05 \times \mathcal{C}$. It follows from this assumption that $\mathcal{C} \geq 20t$. Our algorithms only need the first $20t$ channels so we assume without loss of generality that $\mathcal{C} = 20t$. (Notice, there is nothing special about the constant 0.05—or the other constants used in our upper bounds. We fix specific values only to gain concreteness in the analyses that follow.)

[2] For example, imagine your network is running a MAC protocol with a hard-coded frequency hopping pattern. If an unrelated network nearby happens to run the same MAC protocol, it will end up generating highly-correlated interference. This is more damaging than random interference, but at the same time is not literally malicious.

[3] For example, Bluetooth divides the 2.4 GHz shared spectrum network into 79 channels. Unless N is an exceptionally large overestimate, we can assume that $\mathcal{C} \geq \log N$ for such a configuration.

Algorithm 1. One phase of Pandemic Broadcast Algorithm Prototype, run @ process u

```
                                                              ▷ odd rounds
 1: select a channel uniformly at random, out of the first $20t$ channels.
 2: if $u$ ==source then
 3:     BROADCAST($m$)
 4: else
 5:     LISTEN
 6:     if received $m$ then $broadcaster_u \leftarrow true$
                                                              ▷ even rounds
 7: select a channel uniformly at random, out of the first $20t$ channels.
 8: if $broadcaster_u$ then
 9:     with probability $0.2$ do BROADCAST($m$), otherwise LISTEN
10: else
11:     LISTEN
12:     if received $m$ then $broadcaster_u \leftarrow true$
```

To aid intuition, we begin by explaining a simplified SSB algorithm that we call the *pandemic broadcast prototype* (PBP). This protocol assumes that $\mathcal{C} \geq n/2$ (a strong assumption). We present it for the sake of exposition, as this strong assumption removes several difficulties faced by the more general setting. Once we explain this algorithm we move on to describing and analyzing our main upper bound results, PBA1 and PBA2, which we present as generalizations of the prototype. As mentioned, we will generalize these algorithms to work for more disruption (i.e., larger t) in Section 3.2.

Pandemic Broadcast Prototype. The PBP algorithm (detailed in Algorithm 1), works as follows. In each round, each process is either a *broadcaster* or a *receiver*. Initially, the source is the only broadcaster, but as processes receive the message from a broadcaster, they too become broadcasters. The algorithm divides rounds into *phases*, each consisting of two rounds. In all rounds, all processes choose their channels with uniform independent randomness, out of the first $20t$ channels. In the first round of a phase, only the source broadcasts. In the second round of a phase, each broadcaster decides to broadcast with independent probability 0.2. We prove the following:

Theorem 1. *If $t \leq 0.05 \times \mathcal{C}$ and $\frac{n}{2} \leq \mathcal{C}$, then the pandemic broadcast prototype algorithm solves the SSB problem in $O\big(\frac{t}{n}\log n\big)$ rounds.*

Proof. We consider the execution in three stages. For the first stage, we focus on the first round of the phase, in which only the source broadcasts. As long as at least $\frac{n}{2}$ processes remain as receivers, the probability, p_r, that at least 1 receiver chooses the same channel as the source is bounded as: $1-(1-\frac{1}{\mathcal{C}})^{n/2} \geq 1-e^{-n/2\mathcal{C}}$. Given that $\frac{n}{2} \leq \mathcal{C}$, we have $\frac{n}{\mathcal{C}} \leq 2$. It follows that the above probability is at least $\frac{n}{4\mathcal{C}}$. Now, the channel chosen by source is disrupted with probability at most $\frac{t}{\mathcal{C}} = \frac{1}{20}$. Therefore, in each odd round, with probability at least $\frac{n}{5\mathcal{C}}$, at least one receiver receives the message from the source. Hence, using a Chernoff bound, we get that after $O(\frac{\mathcal{C}}{n}\log n) = O(\frac{t}{n}\log n)$ rounds, number of the broadcasters is $\Omega(\log n)$.

For the second stage, we focus on the even rounds. For this, we prove that as long as number of broadcasters is less than $n/2$, in every $\Theta(\frac{t}{n})$ rounds, the number of the broadcasters doubles with high probability. This, proves that after $O(\frac{t}{n}\log n)$ rounds, the number of broadcasters is at least $\frac{n}{2}$. To this end, consider an arbitrary even round

r and suppose that the set of broadcasters at this round is B_r. For this round, we call a channel *active* if at least one broadcaster selects it. We first show that, with high probability, the number of active channels is at least $\frac{|B_r|}{4}$.

For each channel j, the probability that channel j is *active* is $p^j_{active} = 1 - (1 - \frac{1}{\mathcal{C}})^{|B_r|} \geq 1 - e^{-\frac{|B_r|}{\mathcal{C}}}$. Since $\frac{|B_r|}{\mathcal{C}} \leq \frac{n}{\mathcal{C}} \leq 2$, we get that $p^j_{active} \geq \frac{|B|}{3\mathcal{C}}$. Hence, overall, the expected number of active channels is at least $\frac{|B_r|}{3}$. Moreover, note that for any two channels j_1 and j_2, the two events of respectively j_1 or j_2 being active are negatively correlated. This is because, for instance, given that j_1 is not active, the number of broadcaster distributed over other channels goes up which means that the probability that j_2 is active increases. One can easily generalize this argument and see that for any subset S of channels, the probability of the event that all the channels in S are active together is at most equal to the product of the probabilities of each of the channels in S being active. Hence, because of this generalized form of negative-correlation, Chernoff bound holds in this case [15]. Therefore, since $|B_r| = \Theta(\log n)$, we can use the Chernoff bound and infer that, with high probability, at least $\frac{|B_r|}{4}$ channels are active.

Next, let us call a channel *promising* if it is active and the number of broadcasters which chose it is at most 5. Using a simple pigeon-hole principle, we get that at most $\frac{|B_r|}{5}$ active channels have more than 4 broadcasters. Thus, at least $\frac{|B_r|}{20}$ channels are promising, with high probability.

Now we say a channel is *good* if (a) exactly one broadcaster transmits on it, (b) it is not disrupted, (c) it has at least one receiver process listening to it. For a promising channel j that has $b_j \in [1, 5]$ broadcasters on it, we have: First, the probability that (a) holds, is at least $\frac{1}{5}(1 - \frac{1}{5})^4 > 0.08$. Second, the probability that (b) holds is at least $\frac{\mathcal{C}-t}{t} \geq 0.95$. Finally, the probability that (c) holds is at least $1 - (1 - \frac{1}{\mathcal{C}})^{n-|B_r|} \geq 1 - e^{-\frac{n-|B_r|}{\mathcal{C}}} \geq 1 - e^{-\frac{n/2}{\mathcal{C}}} \geq \frac{n}{4\mathcal{C}}$. Thus, we conclude that every promising channel is *good* with probability at least $\Omega(\frac{n}{\mathcal{C}})$. Moreover, we again have the generalized form of negative correlation. For instance, for two promising channels j_1 and j_2, the events of them being *good* are negatively correlated. This is because, for example if j_1 is not good, then we know that it lacks at least one of properties (a) to (c). But for j_2, (a) holds with probability at least 0.1, independent of what happens on j_1, and also j_1 lacking (b) or (c) just makes j_2 more likely to have (b) or (c), respectively.

Since there are at least $\frac{|B_r|}{20}$ promising channels in round r, we get that, in round r, the expected number of *good* channels is at least $\Theta(\frac{n|B_r|}{\mathcal{C}})$. Similarly, since the number of broadcasters is non-decreasing, we see that in the $\Theta(\frac{\mathcal{C}}{n}) = \Theta(\frac{t}{n})$ even rounds starting with round r, the expected number of good channels is at least $2|B_r|$. Using a Chernoff bound again, which holds because of the aforementioned generalized version of negative correlation, and since $|B_r| = \Omega(\log n)$, we get that after $\Theta(\frac{t}{n})$ rounds, at least $|B_r|$ new broadcasters are recruited, with high probability. In other words, with high probability, the number of broadcasters at least doubles in every $\Theta(\frac{t}{n})$ rounds. This completes the proof of the second stage.

Thus far, we have settled the cases of stages 1 and 2 and we know that after $O(\frac{t}{n}\log n)$ rounds, the number of broadcasters is at least $\frac{n}{2}$. For the third stage, consider an arbitrary process v that remains a receiver by the end of second stage. We show that in $O(\frac{t}{n}\log n)$ rounds after the second stage, v gets the message m with high probability.

For this, similar to above we see that in each even round of third stage, at least $\frac{n}{40}$ channels are promising. Now in each such round, v chooses a channel at random. Thus the probability that this channel is (i) promising, (ii) has exactly one broadcaster, and (iii) is not disrupted is at least $\frac{n/40}{\mathcal{C}} \times \frac{1}{5}(1-\frac{1}{5})^4 \times \frac{\mathcal{C}-t}{\mathcal{C}} = \Theta(\frac{n}{\mathcal{C}}) = \Theta(\frac{n}{t})$. Thus, in $O(\frac{t}{n}\log n)$ even rounds, v has received the message m with high probability. Hence, by a union bound, by that time all the nodes have received it with high probability.

Generalizing the Prototype. We begin by asking what happens when we run PBP for $\mathcal{C} < n/2$. The first stage from our analysis still works, and in $O(\log n)$ rounds, we recruit $\Omega(\log n)$ receivers. In the second stage, however, the doubling process stops when the number of broadcasters passes $\mathcal{C}$, at which point they might start causing collisions, slowing down future recruitment. This creates problems as now, in the third stage of the analysis, the proof breaks down due to this contention.

To solve this problem, it would be sufficient to provide the broadcasters an estimate $\tilde{B}$ of $|B|$, as they could then reduce their broadcast probability to minimize collisions, regardless of their numbers (i.e., reducing down to around $\mathcal{C}$ broadcasters is optimal, as this allows a constant number per channel). An easy way to determine $\tilde{B}$ is to try $\log n$ exponentially growing guesses, one of which would be close to the actual size of B. This approach, however, has a slow-down factor of $\Theta(\log n)$, resulting in a $\Theta(\log^2 n)$ factor in the time complexity—which is too slow.

To avoid this overhead we need more efficient estimation routines. In the next two sections, *we present two algorithms that generalize PBP by implementing efficient broadcaster estimation routines: PBA1 and PBA2.* The PBA1 algorithm assumes $\mathcal{C} > \log n$ and leverages this channel diversity to gain efficiency. The PBA2 algorithm, by contrast, has fewer channels to work with. It leverages the presence of a distinguished source (which breaks symmetry in an important way) to achieve an estimation that is slower than PBA1, but still faster than the $\log n$ overhead of our simple suggestion from above.

Pandemic Broadcast Algorithm 1. As mentioned, the PBA1 algorithm, detailed in Algorithm 2, can be understood as a generalization of PBP. In more detail, we now increase the size of a phase to include the following 5 rounds: The first two rounds are the same as in PBP. In the next two rounds, broadcasters find an estimate of $|B|$ that is in $[\frac{|B|}{4}, 4|B|]$, with at least a nonzero constant probability (described below). We are able to accomplish this in only 2 rounds by moving guesses from $\log n$ consecutive rounds to $\log n$ channels during the same round. In the final round, broadcasters sub-sample themselves using this estimate and then, similar to the second round, broadcast in uniformly chosen channels. We show that $O(\lceil\frac{t}{n}\rceil\log n)$ phases are enough for delivering the message to every process, yielding the total complexity of $O(\lceil\frac{t}{n}\rceil\log n)$ rounds.

The core novelty of PBA1, therefore, is the 2-round estimation subroutine. This routine consists of a *test* and a *report* segment. In the test segment, each broadcaster v chooses one of the channels using an exponential probability distribution. Then broadcaster v, having picked channel f, decides to transmit or listen with probability 0.5. In this *test* segment, any broadcaster that listens to a channel f and receives a message on that channel, estimates $|B|$ to be 2^{f+1}. In the report segment, all broadcasters choose

Algorithm 2. Pandemic Broadcast Algorithm 1, run @ broadcaster process u

```
1: for $phase = 1$ to $\Theta(\log n)$ do
2:     select a channel uniformly at random, out of the first $20t$ channels.    ▷ source's broadcast round
3:     if $u$ ==source then
4:         BROADCAST($m$)
5:     else
6:         LISTEN
7:         if received $m$ then $broadcaster_u \leftarrow true$
8:     select a channel uniformly at random, out of the first $20t$ channels.    ▷ simple relaying round
9:     if $broadcaster_u$ then
10:        with probability $0.2$ do BROADCAST($m$), otherwise LISTEN
11:    else
12:        LISTEN
13:        if received $m$ then $broadcaster_u \leftarrow true$
14:    if $broadcaster_u$ then
15:        $est \leftarrow \mathcal{C}$;  $estimationFlag \leftarrow false$    ▷ test segment of estimation
16:        select random channel $f$ (of source) from an exponential probability distribution
17:        with probability $0.5$ do BROADCAST($m$) , otherwise LISTEN
18:        if received message $m$ then
19:            $est \leftarrow 2^{f+1}$;  $estimationFlag \leftarrow true$
20:        select channel 1 (of source).    ▷ report segment of estimation
21:        if $estimationFlag$ then
22:            with probability $0.05$ do BROADCAST($est$), otherwise LISTEN
23:        else
24:            LISTEN
25:    else    ▷ for receivers to keep them in synch with broadcasters
26:        select a channel uniformly at random, out of the first $20t$ channels.
27:        LISTEN
28:        LISTEN
29:    select a channel uniformly at random, out of the first $20t$ channels.    ▷ final relaying round
30:    if $broadcaster_u$ then
31:        with probability $\min\{\frac{\mathcal{C}}{est}, 0.2\}$ do BROADCAST($m$), otherwise LISTEN
32:    else
33:        LISTEN
34:        if received $m$ then $broadcaster_u \leftarrow true$
```

the same channel, and every broadcaster that made an estimate in the previous segment broadcasts their estimate. Any broadcaster that receives such an estimate adopts it as their est of $|B|$. If a broadcaster learns no estimate, it uses the default value of $\mathcal{C}$.

Intuitively, if $|B| = 2^{f+1}$, then we expect a constant number of processes to choose f, leading to a constant probability of a single process broadcasting and a small number receiving its messages. Because the channel selection probabilities grow exponentially, we can show that the probability that the same happens on other frequencies, sums to a constant. Therefore, with a constant probability, we have a single process reporting an estimate, and the estimate is correct.

A wrinkle here is that the adversary might choose to consistently jam the channel corresponding to the right estimate, or it might jam the reporting channel. To avoid this, we recall that only broadcasters participate in this estimate. Therefore, all participants have received a message from the source. We assume this message can contain sufficiently many bits (or a seed to a pseudo-random number generator) so that in each round of the estimation routine, the broadcasters can shift the channels (circular-shift) by a random amount, unknown to the adversary. In the pseudo-code of Algorithm 2, in the estimation rounds, broadcasters choose their channels using the random shift provided by source in the initial message. [4] Therefore, the probability that these key channels are disrupted is the same as that of a random channel being disrupted. Formally, we prove:

Theorem 2. *If $t \leq 0.05 \times \mathcal{C}$, and $\log n \leq \mathcal{C}$, then the pandemic broadcast algorithm 1 solves the SSB problem $O(\lceil \frac{t}{n} \rceil \cdot \log n)$ rounds.*

Proof (Proof Outline). The analysis of the case where $\mathcal{C} \geq \frac{n}{2}$ is as done in Theorem 1, by focusing only on the first two rounds of each phase, and noticing that in that case, each phase has only 5 rounds. For this case, we proved time complexity of $O(\frac{t}{n} \cdot \log n)$ rounds in Theorem 1. On the other hand, for the case where $\mathcal{C} \in [\log n, \frac{n}{2}]$, we show that in $O(\log n)$ rounds, the message is delivered to every node. For this, we prove the lemmas 1, 2, and 3. Please see the full paper for the proofs. We remark that, the main change, where the effect of estimation part comes in, is studied in Lemma 3.

Lemma 1. *If $t \leq 0.05 \times \mathcal{C}$ and $\mathcal{C} \in [\log n, \frac{n}{2}]$, after $O(\log n)$ phases of PBA1, the number of broadcasters is at least $\Theta(\log n)$.*

Lemma 2. *If $t \leq 0.05 \times \mathcal{C}$ and $\mathcal{C} \in [\log n, \frac{n}{2}]$, after $O(\log n)$ phases of PBA1, the number of broadcasters is at least $\mathcal{C}$.*

Lemma 3. *If $t \leq 0.05 \times \mathcal{C}$ and $\mathcal{C} \in [\log n, \frac{n}{2}]$, after $O(\log n)$ phases of PBA1, all receivers receive the message m, with high probability.*

Pandemic Broadcast Algorithm 2. If $\mathcal{C} \leq \log n$, we no longer can assume that we have at least $\log n$ channels and this prevents us from running the 2-round estimation routine of PBA1. In this case, we use Pandemic Broadcast Algorithm 2 (PBA2). This algorithm is divided into three explicit parts. In the first part, we grow the number of broadcasters to at least $\mathcal{C}$, in $\Theta(\log n)$ rounds, using the PBP strategy. We then stop the recruitment and move on to the second part, where we run a new estimation subroutine that estimates the number of recruited broadcasters to within a factor of 2, with high probability, in $O(\log n \cdot \log\log n)$ rounds (detailed below). In the final part, we use this estimate to sub-sample the broadcasters, by having each broadcaster now broadcast with probability $\min\{\frac{\mathcal{C}}{est}, 0.2\}$. This part runs for $\Theta(\log n)$ rounds, by the end of which the remaining processes have all received the message with high probability. Formally, this gives us the following:

Theorem 3. *If $t \leq 0.05 \times \mathcal{C}$ and $\mathcal{C} \leq \frac{n}{2}$, then the pandemic broadcast algorithm 2 solves the SSB problem in $O(\log n \cdot \log\log n)$ rounds.*

Returning to the algorithm details, notice that the first part of PBA2 is the same as running $\Theta(\log n)$ phases of PBP, and thus its correctness follows from Lemmas 1 and 2. Similarly, the third part is similar to running PBP with the addition that in the second

[4] Because $O(\log n)$ rounds are sufficient to solve the problem, this would require $O(\log n \log t)$ total bits, which could fit under the standard assumption of messages holding a polylogarthmic number of bits.

round of each phase, broadcasters decide to transmit with probability $\min\{\frac{\mathcal{C}}{est}, 0.2\}$, where est is the estimate of $|B|$ to within a factor of 2. An argument similar to that given in proof of Lemma 3 establishes that $\Theta(\log n)$ rounds are sufficient for all remaining receivers to become broadcasters. Therefore, we are left to describe and analyze the estimation part of the algorithm—which is where we turn our attention next.

Estimation Part of PBA2: We change the role of some broadcasters to *mirrors* using the following rule: the processes which received the message in the first $\Theta(\log n)$ rounds of part 1 become *mirrors* while the other processes which received the message remain *broadcasters* [5]. The constants in the asymptotic notations are selected such that, with high probability, at least one mirror exists and there are at least $\frac{\mathcal{C}}{2}$ broadcasters. The core idea is to estimate the number of broadcasters using the help of mirrors.

Theorem 4. *If $t \leq 0.05 \times \mathcal{C}$ and $\mathcal{C} \leq \frac{n}{2}$, and there is at least one mirror, then Estimation Algorithm (presented in Algorithm 4) produces a 2-approximation for the number of broadcasters, with high probability and in $O(\log n \cdot \log\log n)$ rounds.*

In the Estimation Algorithm, our basic tool is a simple probabilistic comparison method called *ApproxCompare*, which compares the number of broadcasters with a given threshold X and outputs a response in form '*larger*' or '*smaller*'. We say that the output is *correct* in case it is 'larger' if the number of broadcasters is greater than $1.4X$, and in case it is 'smaller' if if the number of broadcasters is less than $X/1.4$. On the other hand, if the number of broadcaster nodes is in $[X/1.4,\ 1.4X]$, we do not expect any guarantee from the output. Next, we explain this comparison method and show that its output is *correct* with high probability. But before that, let us finish the story of the estimation algorithm considering a black-box algorithm ApproxCompare. To get a 2-estimation, it is enough to compare the number of broadcasters with thresholds 2^i for $i \in [\log N]$, using ApproxCompare. Moreover, we can use a binary search over these thresholds, to speed up this process. The pseudo-code presented in Algorithm 4 realizes this idea, with some special care about anomalies possible due to incomplete guarantee of the aforementioned definition of correct output. The related correctness and the time-complexity are studied in the proof of Theorem 4.

ApproxCompare Algorithm: As presented in Algorithm 3, *ApproxCompare* procedure is comprised of $200 \log n$ phases. Throughout these phases, both mirrors and broadcasters always work on channel 1 (of the source). Each broadcaster has a variable *counter* initially set to zero. In each *phase*, there are two rounds, namely a *test* round and a *report* round. In the test round, the source transmits message m; also each broadcaster transmits message m with probability $p_X = 1 - 2^{-1/X}$ and remains silent otherwise. In this round, mirrors only listen. Then, in the report round, the source again transmits message m. However this time, each mirror that did not receive a message transmits and broadcasters all listen. Each broadcaster that does not receive a message in the report round increments its counter. After all phases are finished, each broadcaster outputs '*larger*' if its counter is more than half of the number of phases, i.e., if it did not receive anything back in the majority of report rounds. Otherwise, it outputs '*smaller*'.

[5] This change remains in effect for the third part as well.

Algorithm 3. ApproxCompare(X) — at process u

```
1: $counter \leftarrow 0$
2: for $i$:=1 to 200 log $n$ do                                   ▷ $\Theta(\log n)$ phases

3:     if $u == source$ then                                      ▷ Test Round
4:         BROADCAST($m$) on randomly shifted channel 1
5:     else if $broadcaster_u$ then
6:         with probability $p_X = 1 - 2^{-1/X}$ do
7:             BROADCAST($m$) on channel 1 (of source)
8:         otherwise
9:             LISTEN to channel 1 (of source)
10:    else if $mirror_u$ then
11:        LISTEN

12:    if $u == source$ then                                      ▷ Report Round
13:        BROADCAST($m$) on a randomly shifted channel 1
14:    else if $mirror_u$ then
15:        if received a message in test round then
16:            BROADCAST($m$) on channel 1 (of source)
17:        else
18:            LISTEN to channel 1 (of source)
19:    else if $broadcaster_u$ then
20:        LISTEN
21:        if did not received a message then $counter \leftarrow counter + 1$

22: if $counter \geq 100 \log n$ then
23:     return 'larger'
24: else
25:     return 'smaller'
```

Algorithm 4. Estimation Algorithm

```
1: $upperLog \leftarrow \log N$
2: $lowerLog \leftarrow 0$
3: while $upperLog - lowerLog > 1$ do

4:     $midLog = \lfloor \frac{lowerLog+upperLog}{2} \rfloor$
5:     $res_1 \leftarrow ApproxCompare(2^{midLog-1})$
6:     $res_2 \leftarrow ApproxCompare(2^{midLog})$
7:     $res_3 \leftarrow ApproxCompare(2^{midLog+1})$

8:     switch $(res_1, res_2, res_3)$ do
9:         case (*, 'smaller', 'smaller')
10:            $upperLog \leftarrow midLog$
11:        case ('larger', 'larger', *)
12:            $lowerLog \leftarrow midLog$
13:        case ('smaller', *, 'larger')
14:            return $2^{midLog}$
15:        default case:
16:            return $2^{midLog}$

17: return $2^{lowerLog}$
```

Lemma 4. *If $t \leq 0.05 \times \mathcal{C}$ and $\mathcal{C} \leq \frac{n}{2}$, and there is at least one mirror, then each call to ApproxCompare procedure gives a* correct *response with high probability.*

Proof. If the number of broadcasters is greater than or equal to $1.4X$, then in the test round of each phase, the probability that at least one broadcaster transmits is $1-(1-p_X)^{|B|} \geq 1-(1-p_X)^{2X} = 1-2^{-1.4} > 0.62$. If in the test round of a given phase, at least one broadcaster transmits, then the transmission of these broadcasters collides with the transmission of the source and thus, mirrors receive no messages. Hence, in the report round of that phase, mirrors all transmit and therefore, once again due to collision with the source's transmission, broadcasters receive no messages. In such a case, broadcasters increment their counter. Hence, we get that if the number of broadcasters is greater than or equal to $1.4X$, in each test, the counter of each broadcaster is incremented with probability at least 0.62. Using Hoeffding's inequality, we can infer that after $200 \log n$ phases, with high probability, the counter of each broadcaster is greater than $100 \log n$ and therefore, response of the approximate comparison is '*larger*'.

On the other hand, if the number of broadcasters is less than or equal to $X/1.4$, then in the test round of each phase, the probability that no broadcaster transmits is $(1-p_X)^{|B|} \geq (1-p_X)^{X/(2)} = 2^{-1/1.4} > 0.60$. If in the test round of a given phase, no broadcaster transmits, then in that round, the mirrors receive the transmission of the source with probability at least $\frac{\mathcal{C}-t}{\mathcal{C}} \geq 0.95$. In that case, in the report round, no mirrors transmits and therefore, broadcasters receive the transmission of the source again with probability at least $\frac{\mathcal{C}-t}{\mathcal{C}} \geq 0.95$. If all of these events happen, broadcasters do not increment their counter. Hence, we get that if the number of broadcasters is less than or equal to $X/1.4$, in each test, the probability that counter of each broadcaster is incremented is at most $1-0.6 \times 0.95 \times 0.95 < 0.45$. Using Hoeffding's inequality, we can infer that after $200 \log n$ phases, with high probability, the counter of each broadcaster is less than $100 \log n$ and therefore, response of the approximate comparison is '*smaller*'.

Proof (Proof of Theorem 4). First, for the time-complexity analysis, notice that there are $\log n$ comparison thresholds and therefore, the binary search over these threshold values as presented in Algorithm 4 requires just $O(\log \log n)$ comparisons. Since each comparison takes $\Theta(\log n)$ rounds, the total time complexity becomes $\Theta(\log n \cdot \log \log n)$.

Now, we analyse the correctness. Consider an arbitrary turn of the while loop in Algorithm 4. We make three calls to ApproxCompare to take into account the fact that when the number of broadcasters is within a 1.4 factor of the comparison threshold, we do not get any guarantee from Lemma 4. Since $1.4^2 < 2$, at most only one of the three thresholds $2^{midLog-1}$, 2^{midLog}, $2^{midLog+1}$ is within 1.4 factor of the number of broadcasters. Thus, noting Lemma 4, we know that with high probability, the output of at most one of the three calls to ApproxCompare in this turn is not true. The case is clear if all the three responses are true. If only the response of the comparison to $2^{midLog-1}$ is not true, then we know that the number of broadcasters is within a 1.4 factor of $2^{midLog-1}$ and thus, less than 2^{midLog}. In this case, we get a response of 'smaller' from the other two comparisons and therefore, following case presented in line 9 of Algorithm 4, the binary search moves in the correct direction. Similarly, if only the response of the comparison to $2^{midLog+1}$ is not true, then we know that the number of broadcasters is within a 1.4 factor of $2^{midLog+1}$ and thus, greater than 2^{midLog}.

In this case, we get response of 'larger' from the other two comparisons and therefore, following case presented in line 11 of Algorithm 4, the binary search moves in the correct direction. In the last case, if the response to the comparison to 2^{midLog} is not true, then the number of broadcasters is within a 1.4 factor of 2^{midLog}. In this case, Algorithm 4 returns 2^{midLog} as the final estimation, which is clearly a 2-factor estimation. Finally, we know from Lemma 4 that with high probability, no other case happens.

3.2 High-Disruption Regime

In Section 3.1, we presented the PBA1 and PBA2 algorithms, which work when t is not too large compared to $\mathcal{C}$. Here we generalize for any $t < \mathcal{C}$.

Our approach is to use $\Theta\left(\frac{\mathcal{C}}{\mathcal{C}-t}\right)$ rounds to simulate one *abstract* round of PBA1 or PBA2 (in particular, $\frac{3\mathcal{C}}{\mathcal{C}-t}$ rounds will prove sufficient). To simulate abstract round r of one of these low-disruption algorithms, we first let broadcasters choose their channel, and whether or not they broadcast, according to logic of the respective algorithm. They then use *these same fixed choices* for the $\Theta\left(\frac{\mathcal{C}}{\mathcal{C}-t}\right)$ simulation rounds that follow. In each of these simulation rounds, these broadcasters permute their channels using the common random bits from the source message. Therefore, all broadcasters that choose the same channel, will be on the same channel for all $\Theta\left(\frac{\mathcal{C}}{\mathcal{C}-t}\right)$ rounds, but this channel will randomly change from round to round. The receivers can continue to choose random channels on which to receive, throughout this period.

We now prove that this simulation strategy allows PBA1 and PBA2 to work in the high-disruption setting at the cost of slow down factor $\frac{\mathcal{C}}{\mathcal{C}-t}$.

Theorem 5. *The pandemic broadcast algorithm 2, augmented with the simulation strategy, solves the SSB problem in $O(\frac{\mathcal{C}}{\mathcal{C}-t}\log n \cdot \log\log n\})$ rounds, for $\mathcal{C} \leq \log n$ and $t < \mathcal{C}$; the pandemic broadcast algorithm 1, augemented with the simulation strategy, solves the SSB problem in $O(\frac{\mathcal{C}}{\mathcal{C}-t}\lceil\frac{t}{n}\rceil \log n)$ rounds, for $\log n \leq \mathcal{C}$ and $1 \leq t < \mathcal{C}$.*

Proof. Consider abstract round r of one of these SSB algorithms augmented with the simulation strategy. We fix the broadcasters channel and broadcast choices at the beginning of this abstract round, and stick with these choices for the simulation rounds that follow. Assume that these fixed choices include a channel c with either less than or more than 1 broadcaster. In the low-disruption setting, no process would receive a message on c. Notice that the same holds here for the $\Theta\left(\frac{\mathcal{C}}{\mathcal{C}-t}\right)$ channels mapped to c throughout the simulation rounds.

Now assume that these fixed choices include a channel c with exactly one broadcaster. It follows that in our simulation of this abstract round, there will be at least one simulation round where the channel mapped to c is undisrupted, with constant probability. This follows because in each such round, c is disrupted with probability at most $\frac{t}{\mathcal{C}}$. Therefore we experience at least one undisrupted simulation round with probability at least $1 - (\frac{t}{\mathcal{C}})^{\frac{\mathcal{C}}{3\mathcal{C}-t}} = 1 - (1 - \frac{\mathcal{C}-t}{\mathcal{C}})^{\frac{3\mathcal{C}}{\mathcal{C}-t}} \geq 1 - \frac{1}{e^3} > 0.95$.

We are, therefore, in effect simulating our low-disruption algorithms in a new type of network model where the adversary disrupts each channel with some independent disruption probability of no more than 0.05. Though PBA1 and PBA2 were originally analyzed in a model with an arbitrary adversary that jams up to $t \leq 0.05 \times \mathcal{C}$ channels, it is easy to verify that the same arguments work in the more well-behaved model simulated here, in which we fix the interference adversary to choose its $t \leq 0.05 \times \mathcal{C}$ channels randomly. Therefore, the same correctness holds under the same conditions, in exchange for the slow down factor of $\Theta\big(\frac{\mathcal{C}}{\mathcal{C}-t}\big)$ caused by the simulation rounds.

4 Lower Bounds

We now present our lower bounds. For the case where $t = O(n)$, we can prove our solution optimal (within $\log\log n$ factors). This bound focuses on showing that it takes a while for the source to choose a non-disrupted channel (clearly, broadcast cannot complete before the source lands on a non-disrupted channel for the first time). For larger t, the task gets more difficult. In this case, the $\lceil \frac{t}{n} \rceil$ term in our time complexity becomes relevant. Proving this term necessary requires that we bound the behavior of the receivers—a difficult task because they can potentially coordinate in advance of receiving the source message, creating dependencies that thwart straightforward lower bound arguments. Below, we present the lower bound for this difficult case under the assumptions that we are in the low disruption regime and using *regular* algorithms [2, 3]: An SSB algorithm is called *regular* if for each process u, there exists a probability distribution π_u over the channels, i.e., $\pi_u : \{1, 2, \ldots, \mathcal{C}\} \rightarrow [0, 1]$, such that the following holds: as long as u has not received the message, in each round r, process u does not transmit and moreover, it selects the channel to which it listens to using the distribution π_u. Once u receives the message, its behavior is no longer restricted. Note that all our algorithms satisfy this *regularity* assumption. It is unclear whether it is the upper or lower bound that would improve in the absence of these properties. We leave that question as interesting future work. Please see the full paper for the proof.

Theorem 6. *Every solution to the SSB problem requires $\Omega\big(\frac{\mathcal{C}}{\mathcal{C}-t}\log n\big)$ rounds. In the low disruption regime, every* regular *algorithm for the SSB problem also requires $\Omega\big(\frac{\mathcal{C}}{\mathcal{C}-t}\lceil \frac{t}{n} \rceil \log n\big)$ rounds.*

References

1. Bar-Yehuda, R., Goldreich, O., Itai, A.: 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)
2. Daum, S., Gilbert, S., Kuhn, F., Newport, C.: Leader Election in Shared Spectrum Radio Networks. In: Proceedings of the International Symposium on Principles of Distributed Computing (2012)
3. Dolev, S., Gilbert, S., Guerraoui, R., Kuhn, F., Newport, C.: The wireless synchronization problem. In: Proc. 28th Symp. on Principles of Distributed Computing, PODC, pp. 190–199 (2009)
4. Dolev, S., Gilbert, S., Khabbazian, M., Newport, C.: Leveraging Channel Diversity to Gain Efficiency and Robustness for Wireless Broadcast. In: Peleg, D. (ed.) DISC 2011. LNCS, vol. 6950, pp. 252–267. Springer, Heidelberg (2011)

5. Dolev, S., Gilbert, S., Guerraoui, R., Newport, C.: Gossiping in a Multi-channel Radio Network: An Oblivious Approach to Coping with Malicious Interference (Extended Abstract). In: Pelc, A. (ed.) DISC 2007. LNCS, vol. 4731, pp. 208–222. Springer, Heidelberg (2007)
6. Dolev, S., Gilbert, S., Guerraoui, R., Newport, C.: Secure Communication Over Radio Channels. In: Proceedings of the International Symposium on Principles of Distributed Computing (2008)
7. Gilbert, S., Guerraoui, R., Kowalski, D., Newport, C.: Interference-Resilient Information Exchange. In: The Proceedings of the Conference on Computer Communication (2009)
8. Gummadi, R., Wetherall, D., Greenstein, B., Seshan, S.: Understanding and Mitigating the Impact of RF Interference on 802.11 Networks. In: Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM, pp. 385–396 (2007)
9. Jin, T., Noubir, G., Thapa, B.: Zero Pre-Shared Secret Key Establishment in the Presence of Jammers. In: Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (2009)
10. Liu, A., Ning, P., Dai, H., Liu, Y.: USD-FH: Jamming-Resistant Wireless Communication using Frequency Hopping with Uncoordinated Seed Disclosure. In: Proceedings of the IEEE International Conference on Mobile Ad Hoc and Sensor Systems (2010)
11. Liu, A., Ning, P., Dai, H., Liu, Y., Wang, C.: Defending DSSS-Based Broadcast Communication Against Insider Jammers via Delayed Seed-Disclosure. In: Proceedings of the IEEE Annual Computer Security Applications Conference (2010)
12. Liu, Y., Ning, P., Dai, H., Liu, A.: Randomized Differential DSSS: Jamming-Resistant Wireless Broadcast Communication. In: The Proceedings of the Conference on Computer Communication (2010)
13. Meier, D., Pignolet, Y.A., Schmid, S., Wattenhofer, R.: Speed Dating Despite Jammers. In: Krishnamachari, B., Suri, S., Heinzelman, W., Mitra, U. (eds.) DCOSS 2009. LNCS, vol. 5516, pp. 1–14. Springer, Heidelberg (2009)
14. Newport, C.: Distributed Computation on Unreliable Radio Channels. Ph.D. thesis. MIT (2009)
15. Panconesi, A., Srinivasan, A.: Randomized distributed edge coloring via an extension of the chernoff–hoeffding bounds. SIAM J. Comput. 26(2), 350–368 (1997), `http://dx.doi.org/10.1137/S0097539793250767`
16. Pöpper, C., Strasser, M., Čapkun, S.: Jamming-Resistant Broadcast Communication without Shared Keys. In: Proceedings of the USENIX Security Symposium (2009)
17. Popper, C., Strasser, M., Čapkun, S.: Anti-Jamming Broadcast Communication using Uncoordinated Spread Spectrum Techniques. IEEE Journal on Selected Areas in Communications 28(5), 703–715 (2010)
18. Slater, D., Tague, P., Poovendran, R., Matt, B.: A Coding-Theoretic Approach for Efficient Message Verification over Insecure Channels. In: Proceedings of the ACM Conference on Wireless Network Security (2009)
19. Strasser, M., Capkun, S., Popper, C., Čagalj, M.: Jamming-Resistant Key Establishment using Uncoordinated Frequency Hopping. In: IEEE Symposium on Security and Privacy (2008)
20. Strasser, M., Pöpper, C., Čapkun, S.: Efficient Uncoordinated FHSS Anti-Jamming Communication. In: Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (2009)
21. Xiao, L., Dai, H., Ning, P.: Jamming-Resistant Collaborative Broadcast Using Uncoordinated Frequency Hopping. IEEE Transactions on Forensics and Security 7(1), 297–309 (2012)