Efficient Communication in Cognitive Radio Networks*

Seth Gilbert
Natl. University of Singapore
Dept. of Computer Science
Singapore, 117417
gilbert@comp.nus.edu.sg

Fabian Kuhn
University of Freiburg
Dept. of Computer Science
79110 Freiburg, Germany
kuhn@cs.uni-freiburg.de

Calvin Newport
Georgetown University
Dept. of Computer Science
Washington D.C., U.S., 20057
cnewport@cs.georgetown.edu

Chaodong Zheng
Natl. University of Singapore
Dept. of Computer Science
Singapore, 117417
dcszhen@nus.edu.sg

ABSTRACT

Devices in a cognitive radio network use advanced radios to identify pockets of usable spectrum in a crowded band and make them available to higher layers of the network stack. A core challenge in designing algorithms for this model is that different devices might have different views of the network. In this paper, we study two problems for this setting that are well-motivated but not yet well-understood: local broadcast and data aggregation.

We consider a single hop cognitive radio network with n nodes that each has access to c channels. We assume each pair of nodes overlaps on at least $1 \leq k \leq c$ channels.

We first describe and analyze CogCast, a randomized algorithm that solves local broadcast in $O((c/k) \cdot \max\{1, c/n\} \cdot \lg n)$ time, with high probability, by spreading information in an epidemic manner through the network.

We then propose CogComp, a randomized algorithm that solves data aggregation in $O((c/k) \cdot \max\{1, c/n\} \cdot \lg n + n)$ time, with high probability. The CogComp algorithm uses CogCast as a key primitive to establish a spanning tree among the nodes, so that data can be aggregated from leaves to root.

We conclude with a collection of lower bounds that show CogCast is near optimal (in particular, within a $\lg n$ factor) in many cases. These bounds introduce new techniques of potential standalone interest for those concerned with proving fundamental limits in the cognitive radio network setting.

Categories and Subject Descriptors

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

*Research supported by Singapore NUS FRC T1 251RES1404, Ford Motor Company University Research Program, NSF Award CCF-1320279, and ERC Grant No. 336495 (ACDC).

PODC'15, July 21–23, 2015, Donostia-San Sebastián, Spain.
 ISBN 978-1-4503-3617-8 /15/07...$15.00.
DOI: http://dx.doi.org/10.1145/2767386.2767422 .

Keywords

Cognitive radio networks; local broadcast; data aggregation

1. INTRODUCTION

In a traditional wireless network, devices are assigned with one (or a set of) dedicated channel(s) for their network communication. In a *cognitive radio network*, by contrast, each device is equipped with an advanced radio (called a *cognitive radio*) that surveys a crowded spectrum band to identify available fragments. These fragments are partitioned into appropriately sized frequency bands, and are then presented to the higher layers of the stack as a set of abstract communication channels. Cognitive radio networks are increasingly relevant due to two scenarios: allowing secondary users (e.g., mobile wireless devices) to use leftover spectrum in licensed bands assigned to primary users (e.g., television broadcasters) [10, 21, 22], and improving the number of networks that can practically coexist in an unlicensed band.

A core challenge of cognitive radio networks is that network conditions can vary between devices. A pair of devices, therefore, might end up with two different channel sets that overlap in only a limited number of unknown positions. This challenge is bad news for network practitioners, as it complicates their efforts to provide reliable services, but it is good news for distributed algorithm designers, as it generates many practically-motivated, interesting, and tractable theory problems well-suited to our analytical strengths.

Rendezvous and Beyond

The majority of algorithmic results for cognitive radio network models focus on the problem of *deterministic rendezvous.* In its basic form, this problem assumes two devices (called "nodes" below), u and v, have access to channel sets C_u and C_v, respectively. Both sets are of size c and $|C_u \cap C_v| \geq k$, for some $1 \leq k \leq c$. Executions proceed in rounds. In each round, each node can participate only on a single channel in its set. Nodes do not know each other's channel set in advance. The rendezvous problem is solved in the first round that u and v choose some common channel in $C_u \cap C_v$.

Many papers have focused on creating deterministic channel selection schedules that guarantee a rendezvous in bounded time [6, 11, 12, 15, 16, 19]. The best solutions achieve an $O(c^2)$ bound [6, 11]. It is straightforward to show that basic uniform randomized

channel hopping would improve this bound to $O(c^2/k)$ (which is better for non-constant k).[1]

In this paper, we turn our attention to two related problems in a generalization of this model with n nodes where every pair of nodes maintains a channel overlap of size at least k. These problems are equally as useful as rendezvous but much less well-understood. In both cases, we deploy randomization to achieve high efficiency and simple algorithms. We present our results below.

Result #1: Local Broadcast

We begin by studying *local broadcast*: a designated *source* node must disseminate a message to all other $n-1$ nodes in the network. This problem is a key primitive in cognitive radio networks because it allows a source to synchronize the remaining nodes in the network (e.g., perhaps disseminating shared random bits or network configuration parameters). A simple strategy to solve local broadcast is for all nodes to run (randomized) rendezvous with the source transmitting its message in each slot. Given the rendezvous bound from above, this would solve the problem in $O(c^2/k)$ slots (with perhaps an extra $\lg n$ factor if high probability with respect to n is required).

In this paper, we describe and analyze a broadcast algorithm CogCast that solves the problem in $O((c/k)\cdot\max\{1,c/n\}\cdot\lg n)$ slots, with high probability. For $n \geq c$, this time bound reduces to $O((c/k)\cdot\lg n)$ slots, which is a factor of c faster than the straightforward solution.

The CogCast algorithm itself is simple: in each time slot, each node randomly chooses a channel among its channel set, all nodes that know the message broadcast, and all other nodes listen.[2] This algorithm gains its complexity advantage by embracing, like a biological virus, an epidemic spread of information through the network: the more nodes that know the message, the faster it disseminates. The advantage of our algorithm's simplicity is that it is robust: because nodes do the same thing in every slot, it can gracefully handle changes to the network conditions, temporary faults, and so on—suitable properties for such a basic primitive.

Result #2: Data Aggregation

We next turn our attention to a more general problem: *data aggregation*. In this case, each node in the network has some data (e.g., a sensor reading), and a designated source node wants to compute a function based on these data. A solution to this problem can be used to solve many theoretical tasks (e.g., reaching consensus to maintain consistency) as well as practical tasks (e.g., analyzing network condition snapshots to calculate a quality of service metric), and is therefore widely useful.

As before, a straightforward solution would be for each node to once again run basic (randomized) rendezvous. In this case, the source node should listen while the non-source nodes transmit their data. As noted, each non-source node will rendezvous with the source within $O(c^2/k)$ slots. However, if multiple nodes share the same channel during the rendezvous, only one can succeed in its transmission. As n grows, this crowding will also grow. Assuming that the contention resolution is fair, the obvious upper bound for this straightforward strategy is $O(c^2n/k)$.

In this paper, we describe and analyze a new data aggregation algorithm CogComp that instead solves the problem in $O((c/k)\cdot\max\{1,c/n\}\cdot\lg n+n)$ slots, with high probability. For $n \geq c$, this reduces to $O\left(n\lg n/k+n\right)$, which can be significantly faster than the straightforward solution.

The CogComp algorithm operates in multiple phases and uses CogCast as a key subroutine. In more detail, the source begins by broadcasting an initiation message using CogCast. This execution of CogCast constructs a spanning tree among the n nodes, with the source node being the root. The phases that follow coordinate nodes sufficiently so that they can use this tree to aggregate data to the root (i.e., the source node) efficiently. This process is complicated by the lack of a common control channel and the corresponding inability for the aggregation to proceed in a well-synchronized manner: the structure of the tree depends on which processes overlap on which channels.

Result #3: Lower Bounds

A natural follow-up question given our local broadcast solution is whether more complex algorithms could beat CogCast's performance. We answer this question in the negative by proving two lower bounds. First, we prove that $\Omega((c/k)\cdot\max\{1,c/n\})$ slots are necessary to solve local broadcast with constant probability, matching our upper bound within an $O(\lg n)$ factor. This bound, however, requires the assumption that local channel labels are arbitrary (e.g., if u and v share channel q, they might each have a different local label for this channel)—an assumption for which CogCast works. Under the assumption of global channel labels, our third lower bound proves the necessity of $\Omega(c/k)$ slots, (nearly) matching our upper bound in the $c \leq n$ case. The first lower bound leverages arguments based on the properties of randomized matchings generated in bipartite graphs, while the second leverages a careful treatment of expectations. We note that lower bounds are relatively rare to date in the related cognitive radio network literature, especially for randomized algorithms.

2. MODEL

We consider a single-hop cognitive radio network containing n nodes (each with a unique identity) and C channels. We divide time into synchronous *slots* and assume all nodes are activated simultaneously. We assume each node has access to c of the C channels, for some $1 \leq c \leq C$, and that each pair of nodes overlap on at least k channels, for some $1 \leq k \leq c$. Each node knows the value of k and its own set of available channels. However, nodes do not know anything about the channels assigned to other nodes. We assume that this channel assignment is static. (Nevertheless, as we shall later see in the discussion section, CogCast can also tolerate changing (dynamic) channel assignments and provide similar guarantees.)

We assume nodes assign *local labels* to their c channels. These labels are assigned arbitrarily and are not necessarily uniform among nodes: i.e., for a given channel q, it might be labeled as i at node u, and as $j \neq i$ at node v. This assumption highlights the reality that nodes in cognitive radio networks are not dealing with a uniform view of the network or pre-assigned channels (e.g., as discussed in [11].) This assumption strengthens our upper bounds but potentially weakens our lower bounds. (With this in mind, we explicitly address the channel label assumption when presenting our lower

[1]The preference for determinism in this literature is common but should not be seen as a necessity. A main justification in these papers is that determinism guarantees success, whereas randomized solutions have a failure probability. While true in many cases (see discussion section for more details), we argue that these error bounds can be easily tuned in practice to be negligible. Another justification for determinism is that once a pair of nodes swap information, they can calculate each other's schedule going forward, simplifying subsequent rendezvous. The same result can be achieved with randomization; e.g., nodes can swap the seed for a pseudorandom number generator.

[2]A detail we are skipping here but is addressed in the model section is contention. In particular, we assume that if multiple nodes transmit on the same channel simultaneously, *one* message succeeds.

bounds and prove results for both local channel label and global channel label assumptions.)

In each time slot, each node can operate (i.e., broadcast or listen) on one of its c available channels. If only one node broadcasts on a channel, then all nodes listening on that channel will receive the message. For the case of multiple broadcasts, we use a collision model that abstracts away a standard backoff procedure: if multiple nodes send messages concurrently on one channel, then one of these messages—chosen uniformly at random—is received by all nodes that are listening on the channel. Each broadcaster gets feedback as to whether it has succeeded or failed, and failed ones receive the message that was sent.[3] This behavior can be implemented via standard backoff protocols, with poly-logarithmic cost (with respect to n), in almost all reasonable radio network models.[4] Interested readers can refer to the appendix of the full version of this paper for more implementation details.

3. RELATED WORK

Cognitive radio networks is an area under active research. From our point of view, in this domain, researchers usually focus on two sub-areas. In the first sub-area, the goal is to build cognitive radios, with an emphasis on physical layer issues such as detecting spectrum use. (E.g., see [4, 8], also see survey papers from Akyildiz et al. [2], and Yucek et al. [23].) In the second sub-area, researchers assume there exists a low-level cognitive radio providing a simple model in which each node has access to a subset of all channels; the question at hand is to develop algorithms to solve various problems (typically, rendezvous). This paper belongs to the second sub-area, where we focus on solving broadcast and aggregation. We will provide a short review of related work below, along with a discussion of rendezvous in cognitive radio networks.

Broadcast

Broadcasting is a fundamental problem in computer networking. Although it has been extensively studied in the context of traditional wireless networks, there has been limited research addressing it in the cognitive radio network setting.

For example, in a system paper [14], the authors consider a multi-hop cognitive radio network. They construct a "minimal neighbor graph" for each node—which contains the minimum set of channels through which the node can reach all its neighbors—so broadcast can be efficiently conducted. More recently, in a paper by Song et al. [20], by carefully constructing channel hopping sequences, the authors were also able to accomplish broadcast in multi-hop cognitive radio networks. Both of these papers focus on simulations to show the effectiveness of their protocols. In this paper, we focus on an algorithmic approach, guaranteeing successful local broadcast with high probability and analyzing our algorithms for worst-case performance.

Another approach for tackling the broadcast problem is to assume that all necessary information is known in advance (e.g., sets channels that are available to nodes in each slot, sets of overlapping channels that are available to pairs of nodes in each slot), and then focus on developing efficient algorithms which can find good schedules to solve the broadcast problem. Both [17] and [13] belong to this category. In this paper, we focus on assuming minimal a priori environmental knowledge.

[3]In the cognitive radio network community, the collision model is usually stronger: even if multiple nodes are sending messages concurrently on one channel, *all* of these messages will be received by all nodes that are listening on this channel (e.g., [6, 11]).

[4]For example, broadcasting with exponentially decreasing probabilities (e.g., [1, 3]), will ensure a message succeeds with high probability within $O(\log^2 n)$ rounds. Whenever a message succeeds, everyone else receives it and aborts. The only node that does not abort is the node that succeeded, and hence it knows that it succeeded.

Rendezvous

Rendezvous is perhaps the most studied problem today in cognitive radio networks (e.g., see [16] for a short survey). The basic requirement is that each pair of nodes in the network must meet every so often. The performance of rendezvous protocols is usually measured in terms of the time it takes for two nodes to meet. Rendezvous can be used to solve local broadcast: once the source node has met all its neighbors, broadcast is complete.

There are several important earlier papers on rendezvous worth mentioning, including [19] by Shin et al., [15] by Lin et al., and [12] by Gu et al. These algorithms typically have time complexity that is polynomial of the total number of channels.

A more recent paper from Gu et al. [11] presents an algorithm that can guarantee rendezvous between two users in $O(\max\{c_u, c_v\}^2)$ time, where c_u and c_v are the number of channels available to the two users, respectively. The key novelty of this work is that the proposed algorithm only relies on local information. Another recent paper from Chen et al. [6] guarantees rendezvous between any two users in $O(c_u \cdot c_v \cdot \lg\lg C)$ time, where C is the total number of channels. Although authors of [6, 11] have shown that their algorithms are (near) optimal when solving rendezvous, CogCast proves that more efficient algorithms exist when solving local broadcast, as the time complexity of CogCast is only $O((c/k) \cdot \max\{1, c/n\} \cdot \lg n)$.

Data Aggregation

Data aggregation provides a powerful tool for performing generic computation. Although there exist numerous works on aggregation in traditional wireless networks (e.g., see [7] for a survey), we are only able to identify one paper [5] that explicitly targets aggregation in cognitive radio networks. However, the models and approaches employed by [5] are very different from the ones considered in this paper. Lastly, we note here that although rendezvous algorithms can potentially be used to solve data aggregation, CogComp is more efficient in many (if not all) cases.

4. THE COGCAST PROTOCOL

In this section, we will describe the CogCast protocol and prove its correctness. Pseudocode and omitted proofs can be found in the full version of the paper.

Protocol Description

CogCast takes a classic epidemic approach: each node that has previously received the message simply chooses a random channel on which to broadcast the message. (We say that a node is *informed* if it has previously received the message; otherwise it is *uninformed*.) More specifically, each node repeats the following for $\Theta((c/k) \cdot \max\{1, c/n\} \cdot \lg n)$ slots: it chooses a channel uniformly at random from its set of c available channels; if it is uninformed, then it listens; otherwise, it broadcasts.

Analysis

We now show that, after $\Theta((c/k) \cdot \max\{1, c/n\} \cdot \lg n)$ slots, every node is informed. The proof divides into two cases, depending on the relationship between n and c. Here we focus on the $c \leq n$ case (as the analysis for the $c \geq n$ case is similar); the complete proof is in the full version of this paper.

The high-level strategy is similar to traditional epidemic analysis: we divide the protocol execution into two stages, and show that by the end of each stage, the set of informed nodes will reach certain size. However, the unknown underlying channel overlapping pattern complicates detailed analysis. For example, if all the nodes share the same k channels, then finding one overlapping channel is hard, yet each overlapping channel is likely to have many nodes. On the other hand, if every pair of nodes share a distinct set of channels, then finding one overlapping channel is easy, yet each overlapping channel is likely to have only a few nodes. As a result, a key challenge in the analysis is to correctly bound the various probabilities so that they hold despite the underlying pattern of overlap.

We now focus on stage one, which is defined to last as long as there are at most $c/2$ informed nodes. In this stage, the message spreads fast: since the number of informed nodes is low, in each slot, each informed node is likely to meet some uninformed node, yielding a typical exponential doubling process. One potential issue, however, is that two broadcasters may choose the same channel and attempt to inform the same uniformed node. We have to be careful to give "credit" to only one of them. We say that a node u *independently informs* v in a slot if, in that slot: u is informed, v is uniformed, u and v both choose the same channel, and no other informed node chooses the same channel.

We now describe the analysis for stage one in more detail. Assume there are at most $c/2$ informed nodes. Fix an informed node u and a time slot. For channel i, define z_i to be the random variable for the number of uniformed nodes that share channel i with u in this slot. We claim:

CLAIM 1. *When $c \leq n$, if there are at most $c/2$ informed nodes in some fixed time slot, then for an informed node u, the probability that u independently informs an uninformed node is at least $\Theta(1/c^2) \cdot \sum_{i=1}^{C} \min\{z_i, c\}$.*

PROOF. Consider a fixed time slot. Assume there are $n - c/2 \leq x \leq n-1$ uninformed nodes, and $1 \leq n - x \leq c/2$ informed nodes.

For a given informed node u, assume it overlaps the x uninformed nodes on y different channels in total. (Recall that it overlaps each of the uninformed nodes on at least k channels.) Moreover, assume for the i^{th} overlapping channel, there are z_i uninformed nodes that overlap with u on this channel. On the other hand, assume for the i^{th} overlapping channel, there z_i' other informed nodes that overlap with u on this channel. Since there are at most $c/2$ informed nodes, we know $z_i' \leq c/2$.

Based on the above definition, we know in a slot, the probability that u informs at least one uninformed node and no other informed nodes have picked the same channel, is:

$$\begin{aligned}
& \Sigma_{i=1}^{y}(1/c) \cdot (1-(1-1/c)^{z_i}) \cdot (1-1/c)^{z_i'} \\
\geq\ & \Sigma_{i=1}^{y}(1/c) \cdot (1-(1-1/c)^{z_i}) \cdot (1-1/c)^{c/2} \\
\geq\ & \Sigma_{i=1}^{y}(1/c) \cdot (1-(1-1/c)^{z_i}) \cdot (e^{-1}) \\
=\ & \Theta(1/c) \cdot \Sigma_{i=1}^{y}(1-(1-1/c)^{z_i}) \\
\geq\ & \Theta(1/c) \cdot \Sigma_{i=1}^{y}\left(1-(1-1/c)^{\min\{z_i,c\}}\right) \\
\geq\ & \Theta(1/c) \cdot \Sigma_{i=1}^{y}\min\{z_i/c, 1-e^{-1}\} \\
=\ & \Theta(1/c^2) \cdot \Sigma_{i=1}^{y}\min\{z_i, c\}
\end{aligned}$$

This immediately leads to the claim. □

Notice that the summation here depends on the pattern of overlap, i.e., whether the set of shared channels is highly congested or widely distributed. At this point, there are two cases, depending on whether many channels have $\min\{z_i, c\} = z_i$ or $\min\{z_i, c\} = c$. For both cases, we show that:

CLAIM 2. *When $c \leq n$, during the execution of* COGCAST*, if there are at most $c/2$ informed nodes, then for an informed node u, in a slot, $\Theta(1/c^2) \cdot \sum_{i=1}^{C} \min\{z_i, c\} = \Omega(kc)$.*

PROOF. Assume there are $n - c/2 \leq x \leq n-1$ uninformed nodes, and $1 \leq n - x \leq c/2$ informed nodes.

Fix a time slot. For a given informed node u, assume it overlaps with the x uninformed nodes on y different channels. Moreover, assume for the i^{th} overlapping channel, there are z_i uninformed nodes that overlap with u on this channel. Based on this definition, we know $\sum_{i=1}^{y} z_i \geq kx$, as we have assumed each pair of nodes overlap on at least k channels.

To calculate $\sum_{i=1}^{y} \min\{z_i, c\}$, consider two cases: (a) for at most $y - k/2 - 1$ different i, $\min\{z_i, c\} = z_i$; or (b) for at least $y - k/2$ different i, $\min\{z_i, c\} = z_i$. In the first case, we know for at least $k/2 + 1$ different i, $\min\{z_i, c\} = c$. Hence, $\sum_{i=1}^{y} \min\{z_i, c\} \geq (k/2+1) \cdot c = \Theta(kc)$. In the second case, we claim $\sum_{i=1}^{y} \min\{z_i, c\}$ is at least $kx/2 \geq k(n - c/2)/2 \geq kc/4 = \Theta(kc)$. This is because, for each channel that u overlaps with some uninformed nodes, u will overlap with at most x uninformed nodes on that channel. I.e., for any i, we have $z_i \leq x$. Hence, for any $k/2$ overlapping channels, the sum of z_i on those channels is at most $kx/2$. As a result, for the remaining $y - k/2$ overlapping channels, the sum of z_i on those channels is at least $kx - kx/2 = kx/2$. (Recall the sum of all z_i is at least kx.) □

Hence we conclude that during stage one, in a slot, with probability at least $\Omega(k/c)$ an uniformed node u will independently inform an uniformed node. From this we can prove (via a Chernoff bound) that with high probability, within $O((c/k) \cdot \lg n)$ slots, at least $c/2$ nodes will be informed.

From this point onwards, the epidemic doubling process slows down since the number of channels limits the number of new nodes that can be uniquely informed in each slot. As a result, we enter stage two, in which we focus on the uninformed nodes. In particular, with a similar calculation as in Claim 1, we show:

CLAIM 3. *When $c \leq n$, if there are at least $c/2$ informed nodes in some fixed time slot, then for an uninformed node u, the probability that u becomes informed is at least $\Theta(1/c^2) \cdot \sum_{i=1}^{C} \min\{z_i, c\} = \Omega(kc)$.*

As before, the key part here is bounding the summation of the channel distributions. Via a union bound, we see that in $O((c/k) \cdot \lg n)$ slots everyone is informed.

The case where $n \leq c$ proceeds similarly, where stage 1 continues until $n/2$ nodes are informed. In this case, during stage 1 each informed node uniquely informs at least one informed node with probability $\Omega(kn/c^2)$; in stage 2, each uninformed node is informed with probability $\Omega(kn/c^2)$.

Combining the the analysis from both cases, we conclude that:

THEOREM 4. *After executing* COGCAST *for $\Theta((c/k) \cdot \max\{1, c/n\} \cdot \lg n)$ time slots, all nodes will be informed, w.h.p.*

Discussion

Notice, due to the simple randomized structure of the algorithm, there is no need to assume a static fixed channel assignment. As long as each pair of nodes, in each slot, share at least k common channels, the process completes in the same manner. Similarly,

the algorithm itself does not depend explicitly on n or k *except* to determine the running time. In a long-lived system where there is no need to terminate, the algorithm has no dependence on any non-observable system parameters.

5. THE COGCOMP PROTOCOL

In this section, we will describe and analyze the COGCOMP protocol, which solves the data aggregation problem in $O((c/k) \cdot \max\{1, c/n\} \cdot \lg n + n)$ slots. COGCOMP contains four phases. We will first give an overview of the protocol, and then explain each phase in more detail.

Protocol Overview

Executing COGCAST (described in Section 4) constructs an implicit spanning tree among the n nodes, with the source node being the root: each node designates as its parent the first node from which it was informed. (In fact, each node is informed only once, because after that it broadcasts in each slot.) We call this tree the *distribution tree*. The goal of COGCOMP is to aggregate values from the leaves of the tree to the root: each child will pass the aggregated information from its subtree to its parent.

Even after the tree is built (via COGCAST), there are several challenges in aggregating data. First, nodes do not know their positions in the distribution tree: each node initially knows only its parent. Thus, after building the tree in the first phase, we proceed in the second and third phases to exchange "local" information on a node's neighborhood in the tree. By the end of these two phases, each node will know how many direct children it has. The second problem has to do with contention during the aggregation process. Each channel can be used by only one node at a time, but many parent-child pairs may be sharing that same channel. If this contention is not handled carefully, one might imagine being delayed by $\Theta(n/c)$ time (for example) at each level of the distribution tree. Hence, in the fourth phase where we carry out aggregation, we use a coordination mechanism to limit contention.

We now describe and analyze each phase in more detail.

Phase One

In phase one, each node executes the COGCAST protocol. The source node broadcasts a fixed message called INIT. Each node records all its actions in phase one. I.e., for each node, for each slot, a node records whether it broadcasts or listens; if it listens in that slot, it records whether it receives INIT; if it broadcasts in that slot, it records whether its message is successfully transmitted. This yields the following lemma:

LEMMA 5. *After phase one of* COGCOMP*, a distribution tree is constructed among nodes, with the source node as the root, w.h.p.*

Phase Two

We first introduce the notion of an (r, c)-cluster.

DEFINITION 6. *An (r, c)-cluster is the set of nodes that were first informed in slot r on channel c during phase one.*[5]

Each node, except the source, belongs to one (r, c)-cluster. The (r, c) tuple will help nodes to recognize their parent, children, and siblings.

In phase two, each node will determine the size of its cluster. Another goal in phase two is to elect a *mediator* for each channel on which some nodes were informed during phase one. As we shall see in phase four, mediators will help avoid contention among nodes so that the aggregation process can quickly make progress.

We now describe phase two in more detail. Phase two contains n slots. Assume that u is a non-source node that was first informed in slot r on channel c during phase one. In each slot during phase two, u goes to channel c and broadcasts the tuple $\langle u, r\rangle$. Notice, either u's broadcast succeeds, or some other node succeeds (as per the assumption that collisions are resolved at a lower layer of the protocol stack). If u fails, then it checks the information within the succeeded message. In particular, if the sender and u are in same cluster, u will add one to a counter that is initially set to one. Otherwise, if u succeeds, then it remains silent and only listens for the rest of the phase, keeps counting number of nodes in its cluster. By the end of the phase, each node has a count of the number of nodes in the same (r, c)-cluster.

Recall we also elect mediators in phase two. In particular, for a channel c on which some nodes were informed during phase one, let r be the last round in which any node was informed on channel c; the mediator is the node with the smallest identifier in the (r, c)-cluster (i.e., the nodes informed in round r on channel c). For each mediator, it will record the size and slot number (i.e., the r within an (r, c) tuple) of each (r, c)-cluster that was informed (during phase one) on its channel.

The following lemma summarizes the properties of phase two:

LEMMA 7. *After phase two of* COGCOMP*, the following guarantees hold, w.h.p.: (a) for each (r, c)-cluster, each node within the cluster knows the size of the cluster; and (b) for each channel on which some nodes were informed during phase one, one unique mediator is elected for that channel, and this mediator knows the size and slot number of each (r, c)-cluster that were informed (during phase one) on this channel.*

Phase Three

We now introduce the notion of an (r, c)-informer:

DEFINITION 8. *(r, c)-informer is the node which informed the nodes in the (r, c)-cluster during phase one.*

Obviously, for each node that is not the source node, it has only one informer: the informer of its cluster (i.e., its parent in the distribution tree). A node may, however, be the informer for multiple different clusters, as it may have informed different sets of nodes in different slots. The purpose of phase three is to let each node know the size of the clusters for which it is the informer.

Phase three is implemented as the reverse of phase one, i.e., it is a "rewind" of phase one. More specifically, assume phase one took l slots. Then in slot i of phase three, for a node u, it will go to the channel that it used in slot $l - i + 1$ during phase one. Moreover, if u broadcast in slot $l - i + 1$ during phase one and succeeded, then it will listen in this slot and record the information it has heard. Otherwise, if u listened in slot $l - i + 1$ during phase one and was informed (for the first time), then it will broadcast the size of its cluster.

Notice, not every broadcast will succeed, as a cluster may contain multiple nodes. At least one succeeds, however, as is guaranteed by our collision model, and is sufficient to yield the following:

LEMMA 9. *After phase three of* COGCOMP*, for each (r, c)-cluster, the corresponding informer will know that it is the informer of this cluster, and the size of this cluster, w.h.p.*

[5]Since nodes may have different labels for same channel, the channel label inside an (r, c) tuple can be considered as the label from a global oracle's perspective.

Phase Four

In phase four, nodes' values are propagated to the source node: each node will first collect values from its children (in the distribution tree), and then pass them along with its own value to its parent.

Phase four contains multiple *steps*, each of which contains three slots. For a node u, it will stay on the same channel within a step. In a step, a node's role is either a *sender* or a *receiver*: if u is still trying to collect values from its children, then it is a receiver in this step; otherwise, if u is trying to pass values to its parent, then it is a sender in this step. In the following description, if u is a receiver in the current step, then we assume it is the informer of cluster (r, c). Otherwise, if u is a sender, then we assume it is in cluster (r, c).

During phase four, a sender can also be a *mediator*. In particular, if a node u was chosen to be the mediator for a channel during phase two, it runs phase four as a non-mediator until it starts sending values to its parent. From that point on, its mediator duties start. Notice, unlike a normal sender—which can terminate after its values have been sent to the corresponding parent, a mediator needs to continue execution until all nodes that were informed on its channel during phase one have passed their values to their parents.

We now describe each step in more detail.

Slot 1. In the first slot within a step, if u is a receiver or a non-mediator sender, then it will listen on channel c in this slot. Otherwise, if u is a mediator, it will broadcast a slot number r' on channel c, indicating nodes in cluster (r', c) should send values to their parent in the following slot. It chooses r' from the list of (r, c)-clusters for which it is the mediator on channel c in reverse order of r.

Slot 2. In the second slot within a step, if u is a receiver, it will listen. Otherwise, if u is a sender and has heard r' in the previous first slot (or knows r', if u is the mediator) such that $r = r'$, it will broadcast the values it has collected from its children along with its own value, and its identity.

Slot 3. In the last slot within a step, if u is a receiver that is the informer of cluster (r, c), and has heard a message from some node v in cluster (r, c) in the previous second slot, it will broadcast v's identity.

If this message is the last message it needs to collect from cluster (r, c), then after this step, it will start collecting values from the next cluster, in descending order of slot number, or start sending values to its parent if there are no more clusters from which it needs to collect values, or terminate if the node itself is the source node and there are no more clusters from which it needs to collect values.

On the other hand, if u is a non-mediator sender, then it will listen in slot three. If it has heard its own identity, then it will terminate as it knows its message has been delivered to its parent. Finally, if u is a mediator, then it will also listen in slot three. Moreover, if it has heard an identity which is the last identity that needs to send values in cluster (r', c), then after this step, it will let next cluster (in slot number's descending order) start aggregation, or terminate if there are no more clusters need aggregation on this channel.

In the following theorem, we argue that phase four will finish within $O(n)$ slots (and correctly aggregate all data to the source node), which implies the total time of CogComp is $O((c/k) \cdot \max\{1, c/n\} \cdot \lg n + n)$. The high-level strategy we employ in the proof is as follows: Assume the length of phase one is l slots. During phase four, let r_i be the first step after which the values of nodes that were informed in slot $l - i + 1$ during phase one have been aggregated. We prove that $r_i = r_{i-1} + O(k_i)$, where k_i is the size of the largest cluster that was informed in slot $l - i + 1$ during phase one. Hence, we know the total time consumption of phase four is $O(\sum_{i=1}^{l} k_i) \leq O(n)$.

THEOREM 10. CogComp *aggregates data to the root in* $O((c/k) \cdot \max\{1, c/n\} \cdot \lg n + n)$ *slots, w.h.p.*

PROOF. To prove this theorem, we need only to show that since the start of phase four, every node will terminate within $O(n)$ steps (as each step contains $\Theta(1)$ slots); and when the source node terminates, it knows the values of all other nodes.

In the reminder of this proof, for the ease of presentation, we use r_i to denote the first step (during phase four) after which the values of nodes that were informed in slot $l - i + 1$ during phase one have been aggregated (i.e., the values of these nodes' descendants, along with their own values, have been passed to their parents). Here, l is length of phase one. Notice, for $1 \leq i \leq l$, r_i is well defined. For $i = 0$, we define $r_0 = 1$: at the first step of phase four, nodes that were informed in the last slot of phase one are ready to pass their values to their parents. We further define the term *section* i to refer to the steps (during phase four) in which nodes that were informed in slot $l - i + 1$ during phase one are aggregating. I.e., section i refers to the steps between step r_{i-1} and step r_i. Here, $1 \leq i \leq l$.

Key Claim. We first show a key claim regarding phase four which bounds the time it takes to aggregate: for any $1 \leq i \leq l$, $r_i = r_{i-1} + O(k_i)$, where k_i is the size of the largest cluster that were informed in slot $l - i + 1$ during phase one.

We prove the above claim by induction. We first consider the base case, where $i = 1$. If at the beginning of step one, the aggregation for section one is already done, then $r_1 = r_0 = 1$, and we have our base case.

Thus let us assume that at the beginning of step one, the aggregation for section one is *not* yet complete. In particular, consider a channel c on which a node u needs to pass values to its parent v. According to Lemma 7, we know u must be in cluster (l, c). Assume w is the mediator for channel c.

First, notice u must be on channel c. This is because it was informed in the last slot during phase one, hence it must not have informed anyone else in phase one. Hence, from the beginning of phase four, it must be on channel c trying to pass its value to v.

Similarly, we know v must be on channel c as well. This is because, by Lemma 9, v knows it needs to collect values from u (more precisely, from cluster (l, c)), and cluster (l, c) has the largest slot number among all clusters it needs to collect values from. Hence, from the beginning of phase four, it must be on channel c trying to collect values from nodes in cluster (l, c).

Lastly, w must be on channel c too. This is because, slot l is the last slot in phase one, and the existence of cluster (l, c) implies the mediator for channel c must be in cluster (l, c). The fact that slot l is the last slot in phase one also implies w will not need to collect values from other nodes. Hence, from the beginning of phase four, w must be on channel c playing the role of mediator.

Now, according to our protocol, from the beginning of phase four, in slot one of each step, w announces that cluster (l, c) is currently being aggregated. (There cannot exist cluster (l', c) such that $l' > l$. Similarly, for any cluster (l', c) where $l' < l$, it will not be processed until cluster (l, c) is done.) Moreover, after each step, on channel c, one node in cluster (l, c) will pass its value to its parent. Since the size of cluster (l, c) is at most k_1, we know that by the end of step $r_0 + k_1 - 1$ nodes in cluster (l, c) must have passed their values to their parent. This completes the proof for the base case.

We now consider the inductive step. Assume for all $j \leq i - 1$, we have $r_j = r_{j-1} + O(k_j)$. Consider $j = i$. If at the beginning of step r_{i-1}, the aggregation for section i is already done, then $r_i = r_{i-1}$, and we have our inductive step. Assume, then, that at the beginning of step r_{i-1}, the aggregation for section i is *not* done yet. In particular, consider a channel c on which a node u needs

to pass values to its parent v. According to Lemma 7, we know u must be in cluster $(l-i+1, c)$. Assume w is the mediator for channel c.

Firstly, notice u must be on channel c. This is because, by the above assumption, at the beginning of step r_{i-1}, u must have already collected all values from its descendants, and is ready to pass these values (along with its own value) to v. Similarly, we know v must be on channel c as well. This is because, by Lemma 9, v knows it needs to collect values from u (more precisely, from cluster $(l-i+1, c)$), and by the inductive hypothesis, at the beginning of step r_{i-1}, it must have finished collecting values from any cluster (l', c') where $l' > l-i+1$. Hence, at the beginning of step r_{i-1}, it must be on channel c trying to collect values from nodes in cluster $(l-i+1, c)$.

Lastly, w must be on channel c too. This is because, by the inductive hypothesis, at the beginning of step r_{i-1}, for any cluster (l', c') where $l' > l-i+1$, its aggregation process must have already completed. Hence, at the beginning of step r_{i-1}, if w is the informer for any cluster (l', c') where $l' > l-i+1$, it must have already finished collecting values from these clusters. On the other hand, the existence of cluster $(l-i+1, c)$ implies the mediator for channel c must be in cluster $(\hat{l}, c)$ such that $\hat{l} \geq l-i+1$. Thus, w will not be a receiver during the aggregation process of cluster $(l-i+1, c)$. As a result, we know at the beginning of step r_{i-1}, w must be on channel c playing the role of mediator.

Now, according to our protocol, from the beginning of step r_{i-1}, in slot one of each step, w announces that cluster $(l-i+1, c)$ is currently being aggregated. (As, by the inductive hypothesis, for any cluster (l', c) where $l' > l-i+1$, its aggregation process is already completed. Similarly, for any cluster (l', c) where $l' < l-i+1$, it will not be processed until cluster $(l-i+1, c)$ is completed.) Moreover, after each step, on channel c, one node in cluster $(l-i+1, c)$ will pass its value to its parent. Since the size of cluster $(l-i+1, c)$ is at most k_i, we know that by the end of step $r_{i-1}+k_i-1$ nodes in cluster $(l-i+1, c)$ must have passed their descendants' values, along with their own values, to their parent. This completes the proof for the inductive step.

Termination. For a non-source node that is not a mediator, assume it was informed in slot r during phase one. Then according to our protocol and the above analysis, by the beginning of step r_{l-r+1}, it must have terminated. For a non-source node that is a mediator. Assume it is the mediator for channel c, further assume during phase one, slot r was the first slot in which some node were informed on channel c. Then according to our protocol and above analysis, by the beginning of step r_{l-r+1}, it must have terminated. Lastly, for the source node, according to our protocol and above analysis, by the beginning of step r_l, it must have terminated.

Time consumption. According to our above analysis, it is easy to see the time consumption of phase four is bounded by $O(r_l) = O(\sum_{i=1}^{l} k_i)$. Notice, for any node, it will only be counted once in one unique k_i, as, for each non-source node, it belongs to one unique (r, c)-cluster. Therefore, we know $\sum_{i=1}^{l} k_i \leq n$. Hence, the time consumption of phase four is at most $O(n)$ slots. □

Discussion

Before proceeding to the next section, we would like to highlight an advantage of CogComp: small message overhead. In particular, if the nodes' values are used to compute a function that is *associative* (e.g., `min`/`max`, `count`), then each node in the tree can locally compute this function based on the values it has already known (i.e., values from its children and itself), and only pass the outcome (instead of all values) to its parent. This still guarantees the source node will eventually get the correct result, yet the message size can be restricted to $O(\texttt{polylog}(n))$.

Another point to note is that there is a simple $\Omega(n/k)$ lower bound for aggregation: if all the nodes share the same k channels, and each channel can only be used by one node at a time, then it takes $\Omega(n/k)$ slots for every node to reports its data. Thus, our aggregation protocol is near optimal for small values of k (i.e., $O(1)$) when $c \leq n$; and there is room for improvement for larger k.

6. LOWER BOUNDS

In this section, we will derive two lower bounds for the local broadcast problem, considering both local and global channel labels. These lower bounds show that the time complexity of CogCast is near optimal in many cases.

The Local Channel Label Case

In this section, we consider the local channel label model, i.e., each node may assign a different name to each channel. To prove the $\Omega((c/k) \cdot \max\{1, c/n\})$ bound, our strategy is to first identify a simple game that captures the core difficulty of the problem. The structure of the game will allow us to generate a strong lower bound. We then connect the game to the local broadcast problem by a reduction argument. To obtain the needed bound, the game we consider is slightly different when $k \leq c/2$ and $c/2 < k \leq c$. We will first focus on the $k \leq c/2$ scenario.

The $k \leq c/2$ Scenario

In this setting, we consider a game called the (c, k)-*bipartite hitting game*. In this game, we fix $\beta \geq 2$ as a global constant. The input is two integers c, k such that $1 \leq k \leq c/\beta$. Consider two sets of nodes each of size c: $A = \{a_1, a_2, \cdots, a_c\}$ and $B = \{b_1, b_2, \cdots, b_c\}$. Let G be a complete bipartite graph on bipartition (A, B). The game is played between a *player* and a *referee*. At the beginning of the game the referee privately selects a matching M of size k from G. The game then proceeds in rounds. In each round, the player *proposes* an edge e in G. If $e \in M$, the player *wins*. Otherwise, the game moves on to the next round. We allow the player to be an arbitrary probabilistic automaton. (I.e., we place no restrictions on how it uses probabilistic behavior to generate its proposals).

In the lemma below, we show a lower bound for winning the (c, k)-bipartite hitting game. In particular, we first calculate the player's losing probability. Then, through a series of inequalities, we show that within $O(c^2/k)$ rounds, the player's losing probability is larger than $1/2$.

LEMMA 11. *Let $\mathcal{P}$ be a player that wins the (c, k)-bipartite hitting game in $f(c, k)$ rounds with probability at least $1/2$, for some $1 \leq k \leq c/\beta$, and some constant $\beta \geq 2$. It follows that $f(c, k) \geq c^2/(\alpha k) = \Theta(c^2/k)$, where $2 < \alpha = 2(\beta/(\beta-1))^2 \leq 8$.*

PROOF. We consider $\mathcal{P}$'s performance against a referee which chooses each of the k edges in its matching M with uniform and independent randomness. (I.e., it chooses the first edge, removes those endpoints from consideration, then chooses a second edge, and so on.) Assume $\mathcal{P}$ runs for $l = c^2/(\alpha k)$ rounds, where $\alpha = 2(\beta/(\beta-1))^2$. This generates a sequence $P = \{e_1, e_2, \cdots, e_l\}$ of l edge proposals. Notice, for the ease of presentation, we assume the referee makes its choice of edges after the player has generated the l proposals. This is possible as the referee makes its choices independent of the player's behavior.

Define L to be the event that the referee's k edge selections are *not* included in these l proposals. We consider the probability that

L happens. To calculate this probability, define $n_i = (c - i + 1)^2$ to be the number of edges that is available for the referee to select when it is choosing the i^{th} edge for M. The probability that the referee's i^{th} selection is in P is therefore less than or equal to l/n_i. (Notice, this probability is not necessarily equal to l/n_i as the number of edges in P that is eligible for the referee to select might be less than l.) Hence, it follows that the probability that the player does not win with its i^{th} proposal is as least $1 - l/n_i$. Therefore, we have $\mathbb{P}(L) \geq \Pi_{i=1}^{k}(1 - l/n_i)$.

We now lower bound this probability:

$$\begin{aligned}
\mathbb{P}(L) &\geq \Pi_{i=1}^{k}(1 - l/n_i) \\
&\geq \Pi_{i=1}^{k}(1/4)^{l/n_i} \\
&\geq \Pi_{i=1}^{k}(1/4)^{l/n_k} \geq (1/4)^{k \cdot \frac{c^2}{\alpha k} \cdot \frac{1}{(c-k+1)^2}} \\
&> (1/4)^{\frac{c^2}{\alpha(c-k)^2}} = (1/4)^{\frac{1}{\alpha} \cdot \frac{1}{(1-k/c)^2}} \\
&\geq (1/4)^{\frac{1}{\alpha} \cdot \frac{1}{(1-1/\beta)^2}} = (1/4)^{\frac{1}{\alpha} \cdot \left(\frac{\beta}{\beta-1}\right)^2} \\
&= (1/4)^{1/2} = 1/2
\end{aligned}$$

Before proceeding, we will provide some explanations for the second inequality in the above derivation. Firstly, notice that $l/n_i \leq 1/2$ as $l/n_i = \frac{c^2}{kn_i} \cdot \frac{1}{\alpha} = \frac{c^2}{kn_i} \cdot \frac{1}{2} \cdot \left(\frac{\beta-1}{\beta}\right)^2 = \frac{c^2}{2kn_i} \cdot (1-1/\beta)^2 \leq \frac{c^2}{2kn_i} \cdot (1 - k/c)^2 = \frac{c^2}{2kn_i} \cdot \left(\frac{c-k}{c}\right)^2 \leq \frac{c^2}{2k \cdot (c-k+1)} \cdot \left(\frac{c-k}{c}\right)^2 < 1/2k \leq 1/2$. Secondly, notice that it is easy to verify the inequality that for any probability $p \leq 1/2$, we have $1 - p \geq (1/4)^p$. With these two observations, we can obtain the second inequality.

The above result implies: within the first l rounds, the probability that the player wins is $1 - \mathbb{P}(L) < 1/2$. □

We then reduce (c, k)-bipartite hitting to local broadcast by demonstrating a strategy for using a fast broadcast algorithm to create a fast solution to the (c, k)-bipartite hitting game—a relationship that allows our above lower bound to transfer over. Notice, the following lemma does not restrict $k \leq c/2$.

LEMMA 12. *Let $\mathcal{A}$ be an algorithm that solves local broadcast with local channel labels in $g(c, k, n)$ slots in a cognitive radio network, with probability at least $1/2$. Then, one can use $\mathcal{A}$ to construct a player $\mathcal{P}_{\mathcal{A}}$ that wins the (c, k)-bipartite hitting game in $\min\{c, n\} \cdot g(c, k, n)$ rounds, for any $1 \leq k \leq c$, with probability at least $1/2$.*

PROOF. We construct our player $\mathcal{P}_{\mathcal{A}}$ to simulate the n nodes in a network where the $n-1$ initially uninformed nodes share the same channel set which overlaps with the source node's channel set in exactly k locations. Let $A = \{a_1, a_2, \cdots, a_c\}$ be the source node's channel set and $B = \{b_1, b_2, \cdots, b_c\}$ be the channels shared by the remaining $n - 1$ nodes. Since we consider local channel label model, we assume the source node sees each a_i labeled as i, and the other $n - 1$ nodes see each b_i labeled as i.

The first key observation for our simulation is that a k-matching M over the complete bipartite graph with bipartition (A, B) also describes a valid overlap of k channels between the source node and the remaining $n - 1$ nodes (i.e., each $(a_i, b_j) \in M$ corresponds to the source node's local channel i being the same as other nodes' local channel j). The second key observation for our simulation is that in order for $\mathcal{A}$ to solve local broadcast in such a network, there must exist a round in which the source node lands on a shared channel with at least one other node (until this happens, the message starting at the source node makes no progress). When the overlaps are viewed as a matching, this is the same as saying that there must exist a round in which the source node chooses some a_i and some other nodes choose b_j such that $(a_i, b_j) \in M$.

Pulling together these pieces we are ready to describe our simulation and establish its correctness. The player $\mathcal{P}_{\mathcal{A}}$ simulates the source node running with channel set A, the other $n-1$ nodes running with channel set B, and the (unknown to the player) matching M chosen by the referee for this execution defining the k overlapping channels. In each simulated round r, for each simulated non-source node u, $\mathcal{P}_{\mathcal{A}}$ guesses (a_r, b_r^u) if it has not tried this proposal before, where a_r is the channel selected by the source node in this round, and b_r^u is the channel selected by node u in this round. Notice, this implies player $\mathcal{P}_{\mathcal{A}}$ can get at most $\min\{c, n\}$ unique guesses in each simulated round: when $c \leq n$, every channel in B may be picked by some non-source nodes, giving at most c unique guesses; and when $c \geq n$, every non-source node may be able to pick a different channel in B, giving at most n unique guesses.

If none of these guesses win the game, it follows that the source node has failed to land on a shared channel with some other nodes, so $\mathcal{P}_{\mathcal{A}}$ can correctly complete the simulated round by simulating no communication between the source node and other nodes. (If some non-source nodes share a channel, $\mathcal{P}_{\mathcal{A}}$, of course, will simulate their communication correctly.)

As noted, to complete local broadcast, their must be a round in which the source node shares a channel with some other nodes. During that round, $\mathcal{P}_{\mathcal{A}}$'s guesses will win the bipartite hitting game. Because $\mathcal{P}_{\mathcal{A}}$ gets up to $\min\{c, n\}$ unique guesses per simulated round, its time complexity is at most a factor of $\min\{c, n\}$ slower than the time complexity of $\mathcal{A}$. □

Lastly, by combining Lemma 11 and Lemma 12, we immediately have the following result.

LEMMA 13. *For any algorithm, when $1 \leq k \leq c/2$, to solve local broadcast under the local channel label model with probability at least $1/2$, the time consumption is at least $\Omega((c/k) \cdot \max\{1, c/n\})$.*

The $k > c/2$ Scenario

In this setting, we use a similar strategy as the $k \leq c/2$ scenario, but focusing on a different hitting game. In particular, we consider a game called the *c-complete bipartite hitting game*. The input to this game is an integer $c \geq 1$. Consider two sets of nodes $A = \{a_1, a_2, \cdots, a_c\}$ and $B = \{b_1, b_2, \cdots, b_c\}$. Let G be a complete bipartite graph on bipartition (A, B). The game is played between a *player* and a *referee*. At the beginning of the game the referee privately selects a *maximum* matching M in G (M can be interpreted as a bijection from from A to B). The game then proceeds in rounds. In each round, the player *proposes* an edge e in G. If $e \in M$ the player *wins*. Otherwise, it moves on to the next round. We allow the player to be an arbitrary probabilistic automaton.

Again, we first show a lower bound for the c-complete bipartite hitting game.

LEMMA 14. *Let $\mathcal{P}$ be a player that wins the c-complete bipartite hitting game in $f(c)$ rounds with probability at least $1/2$, for some positive constant c. It follows that $f(c) \geq c/3$.*

Now, notice Lemma 12 is also applicable to the $k > c/2$ scenario (as the (c, k)-bipartite hitting game becomes the c-complete bipartite hitting game when $k = c$). This implies there exists a strategy for using a fast broadcast algorithm to solve c-complete bipartite hitting fast. Therefore, by combining Lemma 12, 13, and 14, we can immediately have the following theorem. As can be seen, it proves that under the local channel label model, COGCAST is near optimal (within a multiplicative $O(\lg n)$ factor).

THEOREM 15. *For any algorithm, to solve the local broadcast problem under the local channel label model with probability at least* $1/2$*, the time consumption is at least* $\Omega((c/k)\cdot\max\{1, c/n\})$.

The Global Channel Label Case

In this subsection, we will turn our attention to the global channel label model, in which all nodes assign the same label to a given channel. It is an easier model, and hence yields a stronger lower bound. Of course, CogCast and CogComp work in this setting without any modification, as the global channel label model is just a special case of the local channel label model.

In the following theorem, we show a lower bound for solving local broadcast in the global channel label model. It is derived from calculating the time consumption it will take for the source node to find a channel (among the set of channels that is available to it) on which it overlaps with other uninformed nodes.

THEOREM 16. *For any algorithm, the expected time consumption to solve the local broadcast problem under the global channel label model is at least* $\Omega(c/k)$.

PROOF. Firstly, notice aside from nodes' coins (if the algorithm is randomized), the expectation is over the network setup: (a) for each node, what is the set of channels that is available to it; and (b) for each pair of nodes, what is the set of channels they overlap.

We consider the following setup: there are $C = k + n(c - k)$ channels in total. Among these C channels, k channels are chosen uniformly at random, and all nodes have access to these channels. For the remaining $n(c - k)$ channels, they are partitioned into n disjoint parts (each of which contains $c - k$ channels) uniformly at random. Each node has access to the channels in one part; and different nodes have access to different parts. In this setup, we ensure each node has access to c channels, and each pair of nodes overlap on at least k channels. Therefore, this setup is legitimate.

With the above setup, we begin to prove the lower bound. Notice, for any algorithm $\mathcal{A}$, to accomplish local broadcast, a necessary (but not sufficient) condition is that the source node chooses an overlapping channel in one slot. Therefore, the expected number of slots for $\mathcal{A}$ to accomplish local broadcast is at least the expected number of slots until the source node lands on an overlapping channel (for the first time).

Let X_i be an indicator random variable which takes value one if the source node chooses the i^{th} non-overlapping channel before it chooses any overlapping channel, where $1 \leq i \leq c - k$. We know $\mathbb{P}(X_i = 1) = 1/(k + 1)$, in spite of the strategy employed by $\mathcal{A}$. (Notice, the randomness comes from the unknown network setup, and the algorithm $\mathcal{A}$ itself if it is randomized.) On the other hand, we know the expected number of slots until the source node lands on an overlapping channel (for the first time) is at least $\mathbb{E}(\sum_{i=1}^{c-k} X_i)+1 = \sum_{i=1}^{c-k} \mathbb{E}(X_i)+1 = \sum_{i=1}^{c-k} \mathbb{P}(X_i = 1)+1 = \sum_{i=1}^{c-k} 1/(k + 1) + 1 = (c + 1)/(k + 1) = \Theta(c/k)$. Hence, the expected number of slots for $\mathcal{A}$ to accomplish local broadcast is at least $\Omega(c/k)$, which proves the lemma. □

As can be seen, for the $c \leq n$ case, CogCast is only away from the optimal by an $O(\lg n)$ factor. For the $c \geq n$ case, unfortunately, the gap is bigger. In fact, when $c \gg n$, we can confirm there exist other algorithms that can out-perform CogCast in some cases. For example, consider the following network setup: there are $C = k+n(c-k)$ channels in total, and all pairs of nodes overlap on the same k channels. Imagine an algorithm in which all nodes hop among these channels using the same predefined hopping sequence (e.g., a sequential scan). Under such scenario, in expectation, within $O(C/k)$ slots, all nodes will hop to one of the k overlapping channels and hence complete the broadcast process. Now, if we assume $c = n^2$ and $c = k + 1$, then the simple hopping-together algorithm can solve broadcast in $O(C/k) = O((k + n)/k) = O(1)$ slots (in expectation); whereas CogCast will need $O((c^2/nk)\cdot\lg n) = O(n \lg n)$ slots. By contrast, notice, in the local channel label model, such hopping-together algorithm is not possible.

Intuitively, the above result is not surprising: when most of the channels are overlapping channels and all nodes are hopping together, nodes can quickly find one overlapping channel and hence become informed. On the other hand, in CogCast, each node is independently and randomly hopping among the channels that are available to it. As a result, when $c \gg n$, a lot of time will be consumed before uninformed nodes and informed nodes can land on some common channel.

7. DISCUSSION

In this paper, we have developed a simple yet efficient local broadcast algorithm for cognitive radio networks. Building on the broadcast algorithm, we then developed an algorithm for data aggregation. We have also derived two lower bounds, which prove our broadcast algorithm is near optimal in many cases. These results provide answers for several open questions. We also believe the techniques used in this paper can be helpful when designing or analyzing other algorithms in cognitive radio networks.

One point worth discussing is the correctness guarantee provided by CogCast. As we have mentioned earlier, in the cognitive radio network community, there is a tendency of favoring determinism over randomization, as deterministic algorithms can usually guarantee correctness. However, consider a *dynamic* setting in which, sets of channels that are available to nodes, and sets of channels on which pairs of nodes overlap, can change over time. In such setting, CogCast works as stated, and provides the same guarantees without any modification. (In particular, proof of Theorem 4 is still valid.) Nevertheless, under such model, when $k < c$, it is easy to see that no algorithm exists that can *guarantee* to solve the local broadcast problem within a finite time period, as (even for a randomized algorithm) there is always some possibility that the channel availability will conspire to prevent communication:

THEOREM 17. *Under the dynamic model, when* $k < c$*, no algorithm exists that can guarantee to solve the local broadcast problem within a finite time period.*

Another interesting point worth discussing is the potential close connection between broadcast in cognitive radio networks and *n-uniform jamming resistant broadcast in multi-channel wireless networks*.[6] In the current literature, there only exists limited research (e.g., [9, 18]) which attempts to achieve communication in the presence of an n-uniform jamming adversary. Moreover, these few algorithms all focus on single-channel wireless networks. Understanding the relationship between these two not well understood problems may allow researchers to find more solutions and insights.

According to our preliminary analysis, it turns out that if an algorithm is able to solve local broadcast in *dynamic* cognitive radio networks with only local channel labels, then it is also able to solve

[6]An x-uniform jamming adversary is capable of partitioning n nodes into x disjoint groups, and make jamming decision for each group individually. For example, for a 3-uniform jamming adversary, she may jam channel one for group one, jam channel one and six for group two, and do nothing for group three. Thus, an n-uniform jamming adversary can make jamming decision for each node individually.

n-uniform jamming resistant broadcast in multi-channel wireless networks. More specifically, we have the following theorem:

THEOREM 18. *Assume $\mathcal{N}'$ is a multi-channel wireless network which contains n nodes and c channels. Assume there is an n-uniform jamming adversary Eve in $\mathcal{N}'$ that can jam at most $0 < k < c/2$ channels in each time slot. Assume $\mathcal{N}$ is a dynamic cognitive radio network in which nodes only have local channel labels. Assume there are n nodes in $\mathcal{N}$, each of which has access to c channels, and each pair of nodes overlap on at least $0 < c - 2k < c$ channels. If an algorithm $\mathcal{A}$ exists that can solve local broadcast in $\mathcal{N}$ in $T(n, c, k)$ time with probability p, then there must exist another algorithm $\mathcal{A}'$ that can solve local broadcast in $\mathcal{N}'$ in $T(n, c, k)$ time with probability p.*

This result implies COGCAST can also be used to solve local broadcast in traditional multi-channel wireless networks, even if an n-uniform jamming adversary is present.

References

[1] N. Abramson. The Aloha system: Another alternative for computer communications. In *Proc. of the November 17–19, 1970, Fall Joint Computer Conf.*, AFIPS 70 (Fall), pages 281–285, 1970.

[2] I. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty. A survey on spectrum management in cognitive radio networks. *IEEE Communications Magazine*, 46(4):40–48, 2008.

[3] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. Macaw: A media access protocol for wireless LAN's. *SIGCOMM Comput. Commun. Rev.*, 24(4):212–225, 1994.

[4] D. Cabric, S. Mishra, and R. Brodersen. Implementation issues in spectrum sensing for cognitive radios. In *Conf. Record of the 38th Asilomar Conf. on Signals, Systems and Computers*, volume 1, pages 772–776, 2004.

[5] Z. Cai, S. Ji, J. He, L. Wei, and A. Bourgeois. Distributed and asynchronous data collection in cognitive radio networks with fairness consideration, 2014.

[6] S. Chen, A. Russell, A. Samanta, and R. Sundaram. Deterministic blind rendezvous in cognitive radio networks. In *Proc. 34th Int. Conf. on Distributed Computing Systems (ICDCS)*, pages 358–367, 2014.

[7] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi. In-network aggregation techniques for wireless sensor networks: a survey. *IEEE Wireless Communications*, 14(2):70–87, 2007.

[8] A. Ghasemi and E. Sousa. Spectrum sensing in cognitive radio networks: requirements, challenges and design trade-offs. *IEEE Communications Magazine*, 46(4):32–39, 2008.

[9] S. Gilbert and M. Young. Making evildoers pay: Resource-competitive broadcast in sensor networks. In *Proc. 31st Symp. on Principles of Distributed Computing (PODC)*, pages 145–154. ACM, 2012.

[10] R. Goodwins. Next-generation wireless networks: from gigabit WiFi to white space, 2013.

[11] Z. Gu, Q.-S. Hua, and W. Dai. Fully distributed algorithms for blind rendezvous in cognitive radio networks. In *Proc. 15th ACM Int. Symp. on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 155–164, 2014.

[12] Z. Gu, Q.-S. Hua, Y. Wang, and F. Lau. Nearly optimal asynchronous blind rendezvous algorithm for cognitive radio networks. In *Proc. 10th IEEE Communications Society Conf. on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 371–379, 2013.

[13] S. Ji, R. Beyah, and Z. Cai. Minimum-latency broadcast scheduling for cognitive radio networks. In *Proc. 10th IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 389–397, 2013.

[14] Y. Kondareddy and P. Agrawal. Selective broadcasting in multi-hop cognitive radio networks. In *Proc. IEEE Sarnoff Symposium*, pages 1–5, 2008.

[15] Z. Lin, H. Liu, X. Chu, and Y.-W. Leung. Jump-stay based channel-hopping algorithm with guaranteed rendezvous for cognitive radio networks. In *Proc. 30th Int. Conf. on Computer Communications (INFOCOM)*, pages 2444–2452, 2011.

[16] H. Liu, Z. Lin, X. Chu, and Y.-W. Leung. Taxonomy and challenges of rendezvous algorithms in cognitive radio networks. In *Proc. Int. Conf. on Computing, Networking and Communications (ICNC)*, pages 645–649, 2012.

[17] C. J. Liyana Arachchige, S. Venkatesan, R. Chandrasekaran, and N. Mittal. Minimal time broadcasting in cognitive radio networks. In *Proc. 12th Int. Conf. on Distributed Computing and Networking (ICDCN)*, pages 364–375. 2011.

[18] A. Richa, C. Scheideler, S. Schmid, and J. Zhang. A jamming-resistant mac protocol for multi-hop wireless networks. In *Proc. 24th Int. Symp. on Distributed Computing (DISC)*, pages 179–193. 2010.

[19] J. Shin, D. Yang, and C. Kim. A channel rendezvous scheme for cognitive radio networks. *IEEE Communications Letters*, 14:954–956, 2010.

[20] Y. Song and J. Xie. A distributed broadcast protocol in multi-hop cognitive radio ad hoc networks without a common control channel. In *Proc. 31st Int. Conf. on Computer Communications (INFOCOM)*, pages 2273–2281, March 2012.

[21] The WhiteSpace Alliance. `http://www.whitespacealliance.org/InTheNews.html`, 2015.

[22] WG802.22 - Wireless Regional Area Networks Working Group. 802.22-2011 - IEEE standard for information technology–local and metropolitan area networks–specific requirements–part 22: Cognitive wireless ran medium access control (MAC) and physical layer (PHY) specifications: Policies and procedures for operation in the TV bands, 2011.

[23] T. Yucek and H. Arslan. A survey of spectrum sensing algorithms for cognitive radio applications. *IEEE Communications Surveys Tutorials*, 11(1):116–130, 2009.