Approximation Algorithms for Data Broadcast in Wireless Networks

Rajiv Gandhi, Yoo-Ah Kim, Seungjoon Lee, Jiho Ryu, *Member*, *IEEE*, and Peng-Jun Wan

Abstract—Broadcasting is a fundamental operation in wireless networks and plays an important role in the communication protocol design. In multihop wireless networks, however, interference at a node due to simultaneous transmissions from its neighbors makes it nontrivial to design a minimum-latency broadcast algorithm, which is known to be NP-complete. We present a simple 12-approximation algorithm for the one-to-all broadcast problem that improves all previously known guarantees for this problem. We then consider the all-to-all broadcast problem where each node sends its own message to all other nodes. For the all-to-all broadcast problem, we present two algorithms with approximation ratios of 20 and 34, improving the best result available in the literature. Finally, we report experimental evaluation of our algorithms. Our studies indicate that our algorithms perform much better in practice than the worst-case guarantees provided in the theoretical analysis and achieve up to 37 percent performance improvement over existing schemes.

Index Terms—Ad hoc networking, approximation algorithms, broadcast algorithms, wireless scheduling.

1 Introduction

Network-wide broadcasting is a fundamental operation in wireless networks, in which a message needs to be transmitted from its source to all the other nodes in the network. There may be multiple messages to be broadcasted from multiple sources. Several network protocols rely on broadcasting, for example, information dissemination, service/resource discovery, or routing in multihop wireless networks. Given that key applications of multihop wireless networks include disaster relief and rescue operations, military communication, and prompt object detection using sensors, the design of low-latency broadcasting scheme is essential to meet stringent end-to-end delay requirements for higher-level applications.

Interference is a fundamental limiting factor in wireless networks. When two or more nodes transmit a message to a common neighbor at the same time, the common node will not receive any of these messages. In such a case, we say that collision has occurred at the common node. Interference range may be even larger than the transmission range, in which case a node may not receive a message from its transmitter if it is within the interference range of another node sending a message. Any communication protocol for wireless networks should contend with the issue of interference in the wireless medium.

- *R. Gandhi is with the Department of Computer Science, Rutgers University, Camden, NJ 08102. E-mail: rajivg@camden.rutgers.edu.*
- *Y.-A. Kim is with the National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, 8600 Rockville Pike, Bethesda, MD 20894. E-mail: kimy3@ncbi.nlm.nih.gov.*
- *S. Lee is with AT&T Labs - Research, 180 Park Ave., Florham Park, NJ 07932. E-mail: slee@research.att.com.*
- *J. Ryu is with the Ulsan National Institute of Science and Technology (UNIST), UNIST-gil 50, Eonyang-eup, Ulsan, South Korea 330-708. E-mail: jihoryu@unist.ac.kr.*
- *P.-J. Wan is with the Department of Computer Science, Illinois Institute of Technology, 10 W. 31st Street, Chicago, IL 60616. E-mail: wan@cs.iit.edu.*

Manuscript received 6 July 2009; revised 15 June 2011; accepted 30 June 2011; published online 17 May 2012.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2009-07-0271. Digital Object Identifier no. 10.1109/TMC.2011.162.

One of the earliest broadcast mechanisms proposed in the literature is flooding [1], [2], where every node in the network transmits a message to its neighbors after receiving it. Although flooding is extremely simple and easy to implement, Ni et al. [3] show that flooding can be very costly and can lead to serious redundancy, bandwidth contention, and collision: a situation known as *broadcast storm*. Since then, a large amount of research has been directed toward designing broadcast protocols which are collision free and which reduce redundancy by reducing the number of transmissions. In this paper, we revisit the data broadcast problem and present improved algorithms that guarantee collision-free delivery and achieve low latency.

1.1 Our Contributions

We present algorithms for One-to-All and All-to-All broadcasting problems. In one-to-all broadcast, there is a source that sends a message to all other nodes in the network. In all-to-all broadcast each node sends its own message to all other nodes. Even the one-to-all broadcasting problem is known to be NP-complete [4]. For both problems, we develop approximation algorithms, which improve the previous results.

- For One-to-All Broadcast problem, we present a simple approximation algorithm (Section 4) that achieves a 12-approximate solution, thereby improving the approximation guarantee of 16 due to Huang et al. [5]. Our algorithm is based on the algorithm of Gandhi et al. [4] and incorporates the following two ideas that lead to the improvement: 1) processing the nodes greedily—in nonincreasing order of the number of receivers, and 2) allowing nodes to transmit more than once. The latter is particularly counter-intuitive as one would expect that the latency would increase if a node transmits more than once. Note that in [4] the analysis of their algorithm gives an approximation guarantee that is greater than 400.

- We then consider the ALL-TO-ALL BROADCAST problem and present two algorithms (called CDA and ICDA) with approximation guarantees of 20 and 34, respectively (Section 5), thereby improving the approximation guarantee of 27 by Huang et al. [6]. Our improved result is due to efficient scheduling techniques to collect data and then perform pipelined broadcasting. In ICDA, all nodes are scheduled to participate in transmissions as early as possible. Even though its theoretical bound is weaker than that of CDA, experimental results show that it provides comparable or better performance than CDA, especially in larger networks.
- We study the performance of our broadcast algorithms through simulations under various conditions. Our results indicate that our algorithms perform much better in practice than the worst case guarantees provided. Our algorithms achieve up to 37 percent improvement on end-to-end latency over existing schemes.

2 Related Work

Several techniques have been proposed for broadcasting in wireless networks. In order to reduce the broadcast redundancy and contentions, they make use of nodes' neighborhood information and determine whether a particular node needs to transmit a message [7], [8], [9], [10], [11], [12], [13], [14]. There has been some work on latency-constrained broadcasting in wired networks [15] and some results do exist for radio networks whose models are essentially the same as ours. In particular, Chlamtac and Kutten [16] show that minimum latency broadcast scheduling is NP-Complete for general (nongeometric) graphs. This result does not directly extend to ad hoc networks which are modeled by a restricted class of geometric graphs called disk graphs. Chlamtac and Weinstein [17] gave an algorithm for efficient broadcasting in multihop radio networks. They proved that for arbitrary graphs, the broadcast latency of their schedule is within $O(\ln(N/r)^2)$ times the optimal, where N is the number of network nodes and r is the maximum distance from the source to any other node.

Basagni et al. [18] present a mobility transparent broadcast scheme for mobile multihop radio networks. In their scheme, nodes compute their transmit times once and for all in the beginning. They provide two schemes with bounded latency. These schemes have approximation factors which are linear and polylogarithmic in the number of network nodes. In effect, they assume that the topology of the network is completely unknown. Although their schemes are attractive for highly mobile environments, their approximation factors are far from what is achievable in static and relatively less mobile environments where the broadcast tree and schedule can be computed efficiently.

Gandhi et al. [4] show that minimizing broadcast latency in wireless networks is NP-complete and then present an approximation algorithm for one-to-all broadcasting. Their algorithm simultaneously achieves a constant approximation both for the latency as well as the number of transmissions. However, the approximation guarantee for the latency of their algorithm is greater tan 400. In this work, we modify their algorithm to obtain a 12-approximation ratio, thereby improving their result significantly. Huang et al. [5] obtained a 16-approximation algorithm for one-to-all broadcasting problem. They also present an algorithm with latency at most $R + O(\log R)$ where R is the maximum euclidean hop distance from the source to any node. However, the hidden constant in $O(\log R)$ is not small (>150). Chen et al. [19] also address the problem of minimizing broadcast latency when the interference range is strictly larger than the transmission range. If α is the ratio of the interference range to the transmission range, then for $\alpha > 1$, they give an $O(\alpha^2)$-approximation algorithm. In particular, when $\alpha = 2$, their algorithm achieves a 26-approximation. However, it is not clear how their algorithm behaves when $\alpha = 1$. For all-to-all broadcast problem, Gandhi et al. [4] present a constant approximation algorithm where the constant factor is quite large ($>1{,}000$). Tiwari et al. [20] consider the one-to-all broadcast problem in 3D space. Mahjourian et al. [21] present an approximation algorithm when both interference range and carrier sensing ranges are larger than transmission range. The all-to-all broadcast algorithm by Huang et al. [6] achieves the approximation factor of 27. In this work, we further improve the approximation guarantee for the all-to-all broadcasting.

Hung et al. [22] provide centralized and distributed algorithms for broadcasting and experimental study of their algorithms with respect to collision-free delivery, number of transmissions and broadcast latency. While their centralized algorithm is guaranteed to be collision free, their distributed algorithm is not. They do not provide any guarantees with respect to the number of transmissions and latency of the broadcast schedule. Williams and Camp [23] survey many wireless broadcast protocols discussed above. They provide a neat characterization and experimental evaluation of many of these protocols under a wide range of network conditions.

3 Preliminaries

3.1 Network Model

When the interference range and the transmission range are identical, a wireless network can be modeled as a unit disk graph (UDG), $G = (V, E)$. The nodes in V are embedded in the plane. Each node $u \in V$ has a unit transmission range. Let $|u, v|$ denote the euclidean distance between u and v. Let $D(u)$ denote the neighbors of u in G. A node $v \in D(u)$ iff $|u, v| \leq 1$.

We assume that time is discrete. Since the medium of transmission is wireless, whenever a node transmits a message, all its neighbors hear the message. We assume that every message transmission occupies a unit time slot: i.e., the latency of a single successful transmission is one unit of time. We say that there is a collision at node w, if w hears a message from two transmitters at the same time. In such a case, we also say that the two transmissions interfere at w. A node w receives a message collision free iff w hears the message without any collision.

We also consider the case when the interference range is strictly larger than the transmission range. Let α denote the ratio of the interference range to the transmission range. Consider nodes u and w such that $1 < |u, w| \leq \alpha$. When w

broadcasts a message, even though u will not receive the message correctly (since it is not in $D(w)$), this can prevent node u from receiving a message broadcast from a node in $D(u)$. Thus, for a node u to receive a message collision free, a node in $D(u)$ must transmit the message and no other node within a distance of α from u must transmit the message.

3.2 Problem Statement

We are given a disk graph $G = (V, E)$ and a set of messages $M = \{1, 2, \ldots, m\}$. We also have a set of sources for these messages: $sources = \{s_j | s_j \text{ is the source of message } j\}$. A node can transmit message j only after it receives message j collision free. A *schedule* specifies, for each message j and each node i, the time at which node i receives message j collision free and the time at which it transmits message j. If a node does not transmit a message then its transmit time for that message is 0. The latency of the broadcast schedule is the first time at which every node receives all messages. The number of transmissions is the total number of times every node transmits any message. Our goal is to compute a schedule in which the latency is minimized.

We consider one-to-all and all-to-all broadcasting problems. One-to-all broadcasting is the operation where there is one source node s which has a message to send all other nodes. In all-to-all broadcasting, each node v has its own message $m(v)$ to send all other nodes. Even the one-to-all broadcasting problem is known to be NP-complete [4].

4 One-to-All Broadcast Algorithm

The algorithm takes as input a UDG $G = (V, E)$ and a source node s. The algorithm first constructs a *broadcast tree*, T_b, rooted at s in which if a node u is a parent of a node w then u is responsible for transmitting the message to w without any collision at w. It then schedules the transmissions so that every node receives the message collision free. The two key differences from the algorithm in [4] that lead to a significantly improved approximation guarantee are

1. Processing the nodes in a greedy manner while constructing the broadcast tree.
2. Allowing a node to transmit more than once.

Both these properties are crucial to the proof of Lemma 4.4 which is central to showing that our algorithm yields a 12-approximate solution. Note that in [4] the analysis of their algorithm gives an approximation ratio of at least 400.

The broadcast tree T_b, is constructed as follows: the set of nodes V is partitioned into *primary nodes* P and *secondary nodes* S (these nodes are also referred to as *dominators* and *connectors* in the literature [5]). Let T_{BFS} be the breadth-first search tree rooted at s. Let $L_i, i = 0, 1, 2, \ldots, \ell$, be the set of nodes at level i in the BFS tree. P is a maximal independent set in G constructed by considering one level at a time starting from L_0 in T_{BFS}. The nodes in P form a dominating set in G, i.e., each node in S is within the transmission range of some node in P. The parent-child relationships in T_b are determined as follows: let $P_i = P \cap L_i$ and $S_i = S \cap L_i$ be the set of primary nodes and secondary nodes, respectively, at level i in the BFS tree. At any level i, the algorithm first considers each node $u \in P_i$ in *nonincreasing order of the number of nodes in* $D(u)$ that do not have a parent yet (in [4] the nodes were processed in an arbitrary manner). The children of u in T_b, $C(u)$, are all the secondary nodes in $D(u)$ that do not have a parent when u is considered (line 20 in BROADCASTTREE). After considering all nodes in P_i, the secondary nodes are considered in the same way (i.e., nonincreasing order of the number of nodes in P_{i+1} that do not have a parent) and assigned its children. This algorithm runs in $O(|V|^2)$ time as each node maintains a set of its potential children for lines 19 and 25 and this set needs to be updated when any node is assigned to its parent (lines 22 and 28 in BROADCASTTREE).

```
BROADCASTTREE (G = (V, E), s)
1   P ← P_0 ← {s} // P is the set of primary nodes.
2   T_BFS ← BFS tree in G with root s
3   ℓ ← maximum number of levels in T_BFS
    // s belongs to level 0
4   for i ← 1 to ℓ do
5       L_i ← set of all nodes at level i in T_BFS
6       P_i ← ∅
7       for each w ∈ L_i do
8           if (P ∩ D(w) = ∅) then
9               P_i ← P_i ∪ {w}
10              P ← P ∪ {w}
11      S_i ← L_i \ P_i
12  P_{ℓ+1} ← ∅
13  S ← V \ P
14  for each node u ∈ V do
15      parent(u) ← NIL
16  for i ← 0 to ℓ do
17      P'_i ← P_i
18      while (P'_i ≠ ∅) do
19              u ← node in P'_i with maximum
                    |{w ∈ D(u) | parent(w) = NIL}|
20              C(u) ← {w ∈ D(u) | parent(w) = NIL}
21              for each w ∈ C(u) do
22                  parent(w) ← u
23              P'_i ← P'_i \ {u}
24      while (∃ w ∈ P_{i+1} s.t. parent(w) = NIL) do
25              u ← node in S_i with maximum
                    |{w ∈ D(u) ∩ P_{i+1} | parent(w) = NIL}|
26          C(u) ← {w ∈ D(u) ∩ P_{i+1} | parent(w) = NIL}
27          for each w ∈ C(u) do
28                  parent(w) ← u
29  V_b ← V
30  E_b ← {(u, w) | u = parent(w)}
31  return T_b = (V_b, E_b)
```

In ONE-TO-ALL BROADCAST, the transmissions are scheduled in two phases. In Phase 1, the algorithm schedules transmissions only to the nodes in set (denoted by X) which contains all primary nodes and nonleaf secondary nodes in T_b. In Phase 2, transmissions are scheduled to send the message to all other nodes. *Note that this leads to some nodes transmitting more than once* which is again a significant departure from the algorithm in [4] in which each node transmits the message at most once. The intuition behind this is that it is not necessary to send a message to terminal nodes early as they are not responsible for relaying the message

further. On the other hand, by reducing the number of recipients in the first phase, a node will need to avoid a smaller number of potential conflicts before sending a message to nonterminal nodes, thus reducing the broadcast time. In Phase 1, nodes are considered one level at a time starting from L_0. Only those primary nodes that have a child in X will transmit the message in this phase. Clearly, for any primary node u if $C(u) \neq \emptyset$ and $C(u) \cap X = \emptyset$ then u will transmit the message in Phase 2. At any level L_i, the secondary nodes are scheduled for transmission only after all transmissions of primary nodes in P_i have been scheduled. While scheduling transmissions, the nodes in P_i as well as $S_i \cap X$ are considered in nonincreasing order of the number of their children in T_b. The algorithm then follows a greedy strategy to schedule the collision-free transmissions to nodes in X. Any transmitting node, u, transmits at the minimum time t that satisfies the following collision-free constraints—1) u must have received the message collision free before time t, 2) no node in $C(u) \cap X$ is hearing any transmissions at time t, 3) no node in $D(u) \cap X$ is receiving the message collision-free at time t.

In Phase 2, transmissions are scheduled so that the nodes in $Y = V \setminus X$ receive the message. Nodes are considered one level at a time. For each $v \in Y$, $parent(v)$ is responsible for transmitting the message collision free to v. Since $P \cap Y = \emptyset$, the secondary nodes do not transmit in Phase 2. Any transmitting node, u, transmits at the minimum time t that satisfies the above three collision-free constraints.

ONE-TO-ALL BROADCAST (G, T_{BFS}, T_b, s)

1 **for** each node $u \in V$ **do**
2 $\quad trTime_1(u) \leftarrow 0$ //u's transmission time in Phase 1
3 $\quad trTime_2(u) \leftarrow 0$ //u's transmission time in Phase 2
4 $X \leftarrow P \cup \{w \in S \mid C(w) \neq \emptyset\}$ // set of transmitters
5 $Y \leftarrow V \setminus X$ // set of terminals
6 // Phase 1 - transmitters will receive the message
7 **for** $i \leftarrow 0$ to $\ell - 1$ **do**
8 $\quad P'_i \leftarrow \{u \in P_i \mid \exists\, w \in C(u) \cap X\}$
9 $\quad X_i \leftarrow$ nodes in $P'_i \cup (X \cap S_i)$ with all primary nodes ordered before the secondary nodes and the primary and secondary nodes listed in the order they were chosen in lines 19 and 25 resp. in BROADCASTTREE.
10 **while** $(X_i \neq \emptyset)$ **do**
11 $\quad u \leftarrow$ first node in X_i
12 $\quad I_1(u) \leftarrow \{t \mid \exists\, w \in C(u) \setminus Y$ that hears a message at time $t\}$
13 $\quad I_2(u) \leftarrow \{t \mid \exists\, w \in D(u) \setminus Y$ that receives a message coll-free at time $t\}$
14 $\quad I(u) \leftarrow I_1(u) \cup I_2(u)$ // Interference set of u
15 $\quad trTime_1(u) \leftarrow \min\{t \mid t > rcvTime(u)$ and $t \notin I(u)\}$
16 $\quad$ **for** each $w \in C(u) \setminus Y$ **do**
17 $\quad\quad rcvTime(w) \leftarrow trTime_1(u)$
18 $\quad X_i \leftarrow X_i \setminus \{u\}$
// Phase 2—the terminals will receive the message
19 $Y' = Y$
20 **for** $i \leftarrow 0$ to ℓ **do**
21 $\quad$ **for** each $u \in S_i \cap Y'$ **do**
22 $\quad\quad v \leftarrow parent(u)$
23 $\quad\quad I_1(v) \leftarrow \{t \mid \exists\, w \in C(v) \cap Y'$ that hears a message at time $t\}$
24 $\quad\quad I_2(v) \leftarrow \{t \mid \exists\, w \in D(v)$ that receives a message coll-free at time $t\}$
25 $\quad\quad I(v) \leftarrow I_1(v) \cup I_2(v)$
26 $\quad\quad trTime_2(v) \leftarrow \min\{t \mid t > rcvTime(v)$ and $t \notin I(v)\}$
27 $\quad\quad$ **for** each $u \in C(v) \cap Y'$ **do**
28 $\quad\quad\quad rcvTime(u) \leftarrow trTime_2(v)$
29 $\quad\quad Y' \leftarrow Y' \setminus C(v)$
30 **return** $trTime_1$, $trTime_2$

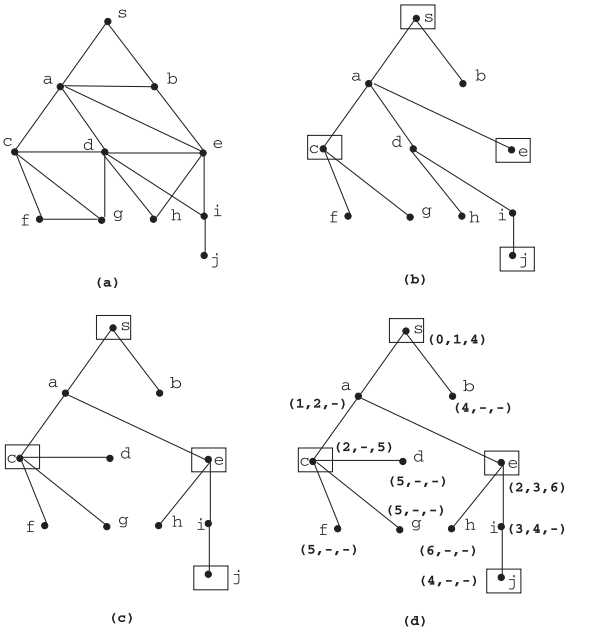

Fig. 1. An illustration of our algorithm. (a) Shows the example network. (b) Shows the BFS tree, T_{BFS} along with the primary nodes (highlighted). (c) Shows the broadcast tree, T_b. (d) Shows the transmission schedule. Besides each node is a 3-tuple, whose members are $rcvTime(\cdot)$, $trTime_1(\cdot)$, and $trTime_2(\cdot)$, respectively. For instance, source node receives a message at time 0 (as it is the original source of the message) and transmits at time 1 for Phase 1 and at time 4 for Phase 2.

Fig. 1 illustrates our algorithm. Note that for any node, the $rcvTime(\cdot)$ that is shown in the figure is the time at which the node is *guaranteed* to receive the message collision free in our algorithm. For example, consider node b. While b receives the message collision free at time 1, in our algorithm it is guaranteed to receive the message collision free at time 4. Similarly, nodes d and h receive message collision free at time 3, but in our algorithm they are guaranteed to receive message collision free at times 5 and 6, respectively. While it is easy to eliminate this slackness from our algorithm, we leave it as is for clarity in exposition.

4.1 Analysis

For any node u, recall that $D(u)$ is the set of nodes in the neighborhood of u. Let $D(u,2)$ ($D_p(u,2)$) be the set of nodes (primary nodes) at a distance of at most 2 from u. We will use the facts that for any node v, the number of primary nodes in $D(v)$, $|D_p(v)| \leq 5$ [5] and $|D_p(v,2)| \leq 19$ [24]. Let $P_j(u)$ denote the primary neighbors of u that belong to L_j.

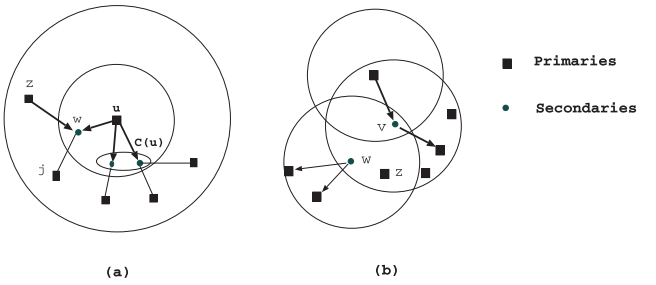

Fig. 2. (a) Proof of Lemma 4.4. An edge between two nodes implies that the nodes are within each other's transmission range. Nodes z and u belong to P_i, and $w \in X \cap C(z)$. z interferes with u at w. Node j, a primary child of w, is guaranteed to exist since $w \in X$. j does not interfere with u since it belongs to $L_{i+1} \cup L_{i+2}$. (b) Proof of Lemma 4.6. For any secondary node w interfering with v, there should be a primary node z in $D(v) \cap D(w)$. There are four cases discussed in Lemma 4.6 and in all cases, $rcvTime(w) \leq t_{i-1} + 8$.

Since our BROADCASTTREE algorithm is an implementation of the algorithm in [4], the properties of T_b proven in [4] hold and we state them in Lemmas 4.1 and 4.2.

Lemma 4.1 ([4]). *T_b is a connected tree rooted at s.*

Lemma 4.2 ([4]). *If $\{u, w\} \in E_b$ then $|\{u, w\} \cap P| = 1$.*

Lemma 4.3. *Consider a node $u \in P_i$, $0 \leq i \leq \ell$. Recall that the set X (line 4 in ONE-TO-ALL BROADCAST) is the set of transmitters. If $C(u) \cap X = \emptyset$ then $trTime_1(u) = 0$.*

Proof. If $C(u) \cap X = \emptyset$, then $u \notin P'_i$ (line 8 in ONE-TO-ALL BROADCAST), hence $trTime_1(u) = 0$. □

The following is our key lemma:

Lemma 4.4. *Let $u \in P_i$. Suppose that in Phase 1, a transmission from u is delayed due to the transmission from a primary node z in $D_p(u, 2) \bigcap P_i$ as z interferes with u at a node w (see Fig. 2a). Then the following is true:*

1. *w is not in $C(u)$.*
2. *For each z, there is at least one unique primary node in $D_p(u, 2)$ that does not interfere with u.*

Proof. The first property is true because the order in which the children of primary nodes are decided in T_b (lines 19-20 in BROADCASTTREE) and the order in which the transmissions are scheduled (line 9 in ONE-TO-ALL BROADCAST) are the same. That is, if there is a primary node z that is scheduled before u and $w \in D(u) \cap D(z)$, then w should be in $C(z)$ and not in $C(u)$.

Now let us consider the second property. Let $w \in D(u)$ be a secondary node at which some node in P_i interferes with u. Clearly, $w \in L_i \cup L_{i+1}$. Since nodes in Y are ignored when computing interference (lines 12 and 13 in ONE-TO-ALL BROADCAST), it must be that $w \in X$. This means that $|C(w)| \geq 1$. Also, since w is a secondary node in $L_i \cup L_{i+1}$, $C(w) \subseteq P_{i+1} \cup P_{i+2}$. Thus, the children of w which are primary nodes in $D_p(u, 2)$ do not interfere with u. This means that for every primary node z in P_i, that interferes with u there is at least one unique primary node in $D_p(u, 2)$ that does not interfere with u. □

In the following, we use Lemma 4.4 and show that even though for any primary node u, $|D_p(u, 2)|$ can be as big as 19 [24], at least half of them are not interfering with u. Let t_i, $0 \leq i \leq \ell$, be the time at which all transmitters in L_i finish transmitting once, i.e., $\forall u \in L_i$, $trTime_1(u) \leq t_i$.

Lemma 4.5. *Consider a node $u \in P_i$, $0 \leq i < \ell$. Let v be a secondary node in $C(u) \cap X$. Then $trTime_1(u) \leq t_{i-1} + \lfloor (17 - |\bigcup_{v \in C(u) \cap X} (P_{i+1}(v) \cup P_{i+2}(v))|)/2 \rfloor + 1 \leq t_{i-1} + 9$.*

Proof. Since $parent(u) \in L_{i-1}$ and all transmitters in L_{i-1} transmit the message by time t_{i-1}, $rcvTime(u) \leq t_{i-1}$. Since the secondary transmitters in L_i are scheduled only after transmitters in P_i, secondary nodes in S_i do not interfere with u. This means that in Phase 1, while scheduling the transmission for u after time t_{i-1}, we must only be concerned about the transmission times of the nodes in P_i that interfere with u and whose transmissions are already scheduled when u is considered. Let $w \in D(u)$ be a secondary node at which some node in P_i interferes with u, which prevents u from transmitting. Then w is not in $C(u)$ by the first property in Lemma 4.4 and $|I(u)| \leq \lfloor (|D_p(u, 2)| - 2 - |\bigcup_{v \in C(u) \cap X} (P_{i+1}(v) \cup P_{i+2}(v))|)/2 \rfloor$, where $I(u)$ (line 14 in ONE-TO-ALL BROADCAST) is the set of times such that for each $t \in I(u)$, a transmission from u at time t would interfere with an already scheduled transmission at time t. The 2 in the numerator accounts for u and $parent(parent(u))$. The 2 in the denominator follows from the second property in Lemma 4.4. Finally, the claim follows because $|D_p(u, 2)| \leq 19$, $|P_{i+1}(v) \cup P_{i+2}(v)| \geq |C(v)| \geq 1$ and since $trTime_1(u) \leq t_{i-1} + |I(u)| + 1$. □

Lemma 4.6. *Let v be a secondary transmitter in L_i, $0 \leq i \leq \ell$. Then $trTime_1(v) \leq t_{i-1} + 12$.*

Proof. If $v \in L_\ell$ then v is not in X. Hence, $trTime_1(v) = 0$. Now assume that $v \in L_i$, $i < \ell$. By Lemma 4.5, we know that all secondary nodes in $S_i \cap X$ receive the message by time $t_{i-1} + 9$. After this time, some secondary nodes in $S_i \cap X$ may interfere with v at primary nodes in $I(v) = P_{i+1}(v) \setminus C(v)$. Therefore, $trTime_1(v) \leq t_{i-1} + 9 + |I(v)| + 1$. Since $|D_p(v)| \leq 5$ and $parent(v) \notin P_{i+1}(v)$, $|I(v)| \leq 3$. If $|I(v)| \leq 2$ then we are done.

Suppose now that $|I(v)| = 3$. Since $|I(v)| = 3$ and $|C(v)| \geq 1$, by Lemma 4.5, $rcvTime(v) \leq t_{i-1} + \lfloor (17 - 4)/2 \rfloor + 1 = t_{i-1} + 7$. Let w be any secondary node in $D(v) \cap S_i \cap X$ (see Fig. 2b). In the following, we will show that for any w, $rcvTime(w) \leq t_{i-1} + 8$ and thus $trTime_1(v) \leq t_{i-1} + 8 + |I(v)| + 1 \leq t_{i-1} + 12$.

First observe that $D(w)$ and $D(v)$ must share a primary node, otherwise in $D(v)$ there will be more than five nodes (three in $I(v)$, at least one in $C(v)$, $parent(v)$ and w) with pairwise distance of greater than 1, which is not possible [5]. Let $z \in D_p(w) \cap D_p(v)$. We consider the four different cases:

1. *z is $parent(v)$ but not $parent(w)$*: w receives the message (strictly) earlier than v as $parent(w)$ is scheduled before $parent(v)$. Therefore, $rcvTime(w) \leq t_{i-1} + 8$.
2. *z is $parent(v)$ and $parent(w)$*: since both v and w are in X, we have $|P_{i+1}(v)| \geq |C(v)| \geq 1$ and $|P_{i+1}(w)| \geq |C(w)| \geq 1$. By Lemma 4.5, we have $rcvTime(w) \leq t_{i-1} + 8$.

3. *z is in $P_{i+1}(v)$ but not in $C(w)$*: since $|C(w)| \geq 1$ (as w is in X), we get $|P_{i+1}(w)| \geq 2$. Plugging this value in the expression in Lemma 4.5, we get $rcvTime(w) = trTime_1(parent(w)) \leq t_{i-1} + 8$.
4. *z is in $P_{i+1}(v) \cap C(w)$*: if $z \in C(w)$ then in line 25 of BROADCASTTREE w must be chosen before v. At that point in the algorithm v would have at least two primary nodes (z and $C(v)$) in $D(v)$ which don't have a parent. This means that $|C(w)| \geq 2$ and again by Lemma 4.5, $rcvTime(w) \leq t_{i-1} + 8$.

Thus, when $|I(v)| = 3$ all secondary nodes in $D(v) \cap S_i \cap X$ (including v) receive the message by time $t_{i-1} + 8$. Hence, $trTime_1(v) \leq t_{i-1} + 8 + |I(v)| + 1 \leq t_{i-1} + 12$. □

Lemma 4.7. *For $0 \leq i \leq \ell - 1$, the time by which all transmitters in L_i transmit the message once is $t_i \leq t_{i-1} + 12$.*

Proof. Follows from Lemmas 4.3, 4.5, and 4.6. □

We now analyze the transmission times for Phase 2.

Lemma 4.8. *For a secondary node $v \in S$, $trTime_2(v) = 0$.*

Proof. In Phase 2, the nodes that transmit the message have a child in Y. Combining Lemma 4.2 with the facts that $C(v) \subseteq P$ and $P \cap Y = \emptyset$ we get $trTime_2(v) = 0$. □

Lemma 4.9. *For $0 \leq i \leq \ell - 2$, let $u \in S_i$ be a terminal node, i.e, $u \in S_i \cap Y$. Then $rcvTime(u) \leq t_{i+2} + 19$.*

Proof. Let $v = parent(u)$. Note that $v \in P_j$, $j \in \{i-1, i\}$. The nodes that may interfere with v belong to the set $D(v, 2)$. Let $B_1(v) = D(v, 2) \cap (L_{j-2} \cup L_{j-1} \cup L_j)$ and $B_2(v) = D(v, 2) \cap (L_{j+1} \cup L_{j+2})$. Thus,

$$|I(v)| \leq |\{(trTime_1(w), trTime_2(w)) \mid w \in B_1(v)\} \cup \{trTime_1(w) \mid w \in B_2(v)\}|.$$

We know that for any node $w \in B_1(v)$, $trTime_1(w) \leq t_i$. By Lemma 4.8, $trTime_2(w) = 0$ for all $w \in S$. Hence, after time t_i, v is guaranteed a collision-free second transmission if v avoids transmitting at $trTime_2(w)$ for each primary node in $w \in B_1(v) \setminus \{v\}$ and at $trTime_1(w)$ for each $w \in B_2(v)$. Thus after time t_i, v must avoid 1) at most one time corresponding to each primary node in $D_p(v, 2) \setminus \{v\}$, cardinality of which is at most 18, and 2) at most one time ($trTime_1(\cdot)$) for each secondary node $w \in D(v, 2) \cap (S_{i+1} \cup S_{i+2})$. There are at most $(t_{i+1} - t_i) + (t_{i+2} - t_{i+1})$ times when all secondary nodes in $S_{i+1} \cup S_{i+2}$ transmit the message. Hence $trTime_2(v) \leq t_i + 18 + (t_{i+1} - t_i) + (t_{i+2} - t_{i+1}) + 1 \leq t_{i+2} + 19$. □

Lemma 4.10. *For any $v \in X$, $rcvTime(v) \leq t_{\ell-1}$.*

Proof. Nodes in $X \cap L_i$, $0 \leq i \leq \ell - 1$, receive their message by the time t_i. Note that $X \cap L_\ell = P_\ell$ and these nodes have their parents in $L_{\ell-1}$. Hence, they receive their message by time $t_{\ell-1}$. □

Lemma 4.11. *Let v be any vertex in L_i, $0 \leq i < \ell - 2$. Then $rcvTime(v) \leq t_{\ell-1} + 19$.*

Proof. If $v \in X$ then by Lemma 4.10, $rcvTime(v) \leq t_{\ell-1}$. If $v \in S$ then by Lemmas 4.6 and 4.9, and the fact that $i < \ell - 2$, we get $rcvTime(v) \leq t_{i+2} + 19 \leq t_{\ell-1} + 19$. □

Lemma 4.12. *For $\ell - 2 \leq i \leq \ell$, for any $u \in L_i$, $rcvTime(u) \leq t_{\ell-1} + 19$.*

Proof. If $u \in X$ then by Lemma 4.10, $rcvTime(u) \leq t_{\ell-1}$. Otherwise, if $u \in Y$, let $v = parent(u)$. Since $trTime_1(w) \leq t_{\ell-1}$ for all $w \in V$, and secondary nodes transmit at most once (Lemma 4.8), v is guaranteed collision-free second transmission after $t_{\ell-1}$ if v avoids second transmission times of nodes in $D_p(v, 2) \setminus \{v\}$. Hence, $trTime_2(v) \leq t_{\ell-1} + |D_p(v, 2)| - 1 + 1 \leq t_{\ell-1} + 19$. □

Theorem 4.13. *Our algorithm gives a 12-approximate solution for the latency. The number of transmissions in our algorithm is at most 21 times those in an optimal solution.*

Proof. Let $v \in V$. From Lemmas 4.11 and 4.12 $rcvTime(v) \leq t_{\ell-1} + 19$. Using Lemma 4.7 we get

$$rcvTime(v) \leq t_1 + 12(\ell - 2) + 19. \quad (1)$$

We know that $t_0 = 1$. For each secondary node $w \in L_1$, secondary nodes at a distance of at most 2 from w can interfere at most three primary nodes in $D(w)$. Hence, $t_1 \leq t_0 + 3 + 1 = 5$. Combining this with (1), we get $rcvTime(v) \leq 12(\ell - 2) + 24 = 12\ell$. Since ℓ is a lower bound on an optimal solution we get a 12-approximate solution. □

Lemma 4.14. *The number of transmissions in our algorithm is at most $3|P|$ where $|P|$ is the size of the primary set.*

Proof. Each primary node transmits at most twice, at most once in Phase 1 and at most once in Phase 2. Since secondary nodes do not transmit in Phase 2, each secondary node transmits at most once. By Lemma 4.2, each secondary node that transmits has a unique child that is a primary node. Hence, the number of transmitting secondary nodes is at most $|P|$. Hence, the total number of transmission is at most $3|P|$. □

Lemma 4.15 ([4]). *The number of transmissions in any optimal algorithm is at least $|P|/7$.*

The above lemmas imply the following theorem:

Theorem 4.16. *The number of transmissions in our algorithm is at most 21 times those in an optimal solution.*

4.2 General Interference Range Model

Our algorithm and analysis can be easily extended to the case when the interference range of a node is different than its transmission range. The only changes are in lines 13 and 24 of the pseudocode ONE-TO-ALL BROADCAST, where $D(u)$ (line 13) and $D(v)$ (line 24) are to be replaced by "interference range of u" and "interference range of v," respectively. Note that if α, the ratio of interference range to the transmission range is a constant, then so is $|D_p(v, \alpha + 1)|$ (note that $|D_p(v)|$ remains the same). The rest of the analysis is very similar to the case when $\alpha = 1$; only the values of some constants will change.

Theorem 4.17. *If the ratio of the interference range to the transmission range of a node, α is bounded by a constant, our algorithm yields a constant factor approximation for latency and the number of transmissions.*

5 All-to-All Broadcast Algorithm

We now consider all-to-all broadcasting, in which each node v has a message $m(v)$ to send to all other nodes. We present two algorithms. By adopting efficient scheduling scheme for pipelined broadcasting, the first algorithm achieves an approximation guarantee of 20, which improves the previous best known guarantee of 27 in the literature [6]. The second algorithm achieves an approximation factor of 34, which performs well in our experiments (Section 6).

A Lower Bound. Let G be a unit-disk graph with n nodes. Denote by γ_c the connected domination number of G. That is, $\gamma_c = |CD(G)|$ where $CD(G)$ is a minimum size connected dominating set of G. Then we have the following *lower bound* on the latency of all-to-all broadcasting.

Lemma 5.1. *The minimum latency of all-to-all broadcasting in G is at least $n - 1 + \gamma_c$.*

Proof. The broadcasting of each message requires at least γ_c transmissions. So, the total number of transmissions in any all-to-all broadcast schedule is at least $n\gamma_c$. This implies that some node must take at least γ_c transmissions. On the other hand, every node must take $n - 1$ receptions. Therefore, some node takes at least $n - 1 + \gamma_c$ transmissions or receptions. This implies that $n - 1 + \gamma_c$ is a lower bound on the minimum latency of all-to-all broadcasting. □

5.1 Collect-and-Distribute Algorithm (CDA)

The *graph radius* of G with respect to a node v is the maximum depth of the BFS tree rooted at v. A *graph center* of G is a node in G with respect to which the graph radius of G is the smallest. Let s be a graph center of G, and R be the graph radius of G with respect to s. Clearly, $\gamma_c \geq R$. We call transmissions of message m from a node v *upward* if the message m is originated from the descendant of v. Otherwise, a transmission is called *downward*. Our schedule consists of two phases. In Phase 1, s collects all the packets by performing upward transmissions. In the Phase 2, s broadcasts all the n packets to all other nodes via downward transmissions.

Phase 1. Node s collects all messages by using the data collection algorithm based on the one by Florens and McEliece [25]. We simplify their algorithm as follows: first construct a BFS tree from s, and sort messages $m(v)$ in nondecreasing order of the level of v in the BFS tree. That is, messages that are closer to s appear first in the sorted list. Let us assume that message j be the jth message in the sorted order. We now greedily schedule transmissions by giving priority to message j over any message $i > j$. The latency of the collection algorithm is at most $3(n - 1)$ [25].

Phase 2. We construct a broadcast tree T_b using BROADCASTTREE in Section 4. Next, we describe transmission scheduling algorithm. In the algorithm by Gandhi et al. [4], the root node collects all messages and perform one-to-all broadcasting for each message. The root node needs to wait until the previous message reaches L_3 before initiating a broadcast for another message to make sure there are no collisions in their algorithm. In our algorithm, we find a schedule by a vertex coloring, which makes sure that all the nodes with the same color can broadcast a message without collision, and show that 17 colors are enough to obtain a collision-free schedule.

Let H_1 (resp., H_2) be the graph over the primaries (resp., secondaries) in which there is an edge between two primaries (resp., secondaries) if and only if one of them has a child adjacent to the other in G.

The scheduling for H_1 can be done by computing a vertex coloring of H_1 in the first-fit manner in the smallest-degree-last ordering. By proper renumbering of the colors, we assume that s has the first color. Let k_1 be the number of colors used by this coloring. Then, $k_1 \leq 12$ [5].

We compute a vertex coloring of H_2 in the first-fit manner by considering nodes in the same order as used when computing the broadcast tree T_b, and let k_2 be the number of colors. Then, $k_2 \leq 5$ [5].

Let $k = k_1 + k_2$. We define a *superstep* to be a group of consecutive k time slots. In each superstep, the first k_1 slots are for scheduling transmissions from primaries, and the remaining k_2 slots will be for secondaries. Each primary (resp., secondary) with color i is only allowed to transmit in the ith slot of a primary (resp., secondary) slot in a superstep. The source node s transmits one packet in each superstep. Each secondary receiving a packet in a superstep transmits the received packet in the corresponding secondary slot in the *same superstep*. Each primary node receiving a packet in a secondary slot transmits the received packet in a primary slot of the *subsequent superstep*. Note that any message that the primaries at level i received in a given superstep will be forwarded to the primaries at level $i + 1$ or $i + 2$ in the next superstep. Therefore, a message which has been sent from a source will be broadcasted to all nodes within R supersteps where R is the number of levels in the BFS tree.

Lemma 5.2. *The second phase takes no more than $17(n - 1 + R)$ time steps.*

Proof. We show that in $n - 1 + R$ supersteps, all n messages are broadcast. After n supersteps, the source node transmits the last packet. After another $R - 1$ supersteps, the last packet reaches all nodes. Hence, the latency of the second schedule requires at most $(n - 1 + R)$ supersteps. As each superstep consists of 17 time slots, we have the lemma. □

Theorem 5.3. *Our all-to-all broadcasting algorithm gives a 20-approximation.*

Proof. Recall that the first phase takes at most $3(n - 1)$ time slots. The second phase takes no more than $17(n - 1 + R)$ time steps as in $n - 1 + R$ supersteps, all n messages are broadcasted and each superstep consists of 17 time slots. Therefore, the total latency of our all-to-all broadcast schedule is at most $3OPT + 17(n - 1 + R) < 20OPT$. □

5.2 Interleaved Collect-and-Distribute Algorithm (ICDA)

In the 20-approximation algorithm proposed in Section 5.1, all messages are first sent to the root node s in the broadcast tree, and then s sends the messages one by one. Note that in the early stages of the algorithm, until s receives all the messages and starts propagating them, most nodes are idle, thus increasing the broadcast time significantly. We now describe an algorithm in which all nodes participate in broadcasting as soon as possible so as to minimize the broadcast time. The main idea is as follows: suppose that a node v receives a message $m(x)$ forwarded originally from

its descendant x in the broadcast tree and relays it to its parent to deliver the message to the root node s. Note that the children of v can also receive the message when v broadcasts it and therefore, they can initiate broadcasting $m(x)$ in their own subtrees in parallel without waiting for the message forwarded from s.

Using the broadcast tree constructed as in CDA, we schedule transmissions for each node as follows: as in CDA, we define a *superstep* but in a slightly different way. That is, in each superstep, every node transmits at most one message (either upward or downward) if there is any message that the node received but not sent. Instead of finishing all upward transmissions first, we mix upward or downward transmissions in each superstep with preferences given to upward transmissions. Also for an upward transmission, a node should make sure that its parent and *all of its children* (except the one which sent the message to v) receive the message. For a downward transmission, v is responsible for sending the message to all of its children. The scheduling algorithm is as follows:

1. **Transmissions from terminal nodes.** Before starting supersteps, all *terminal nodes* Y send messages to their parents one by one.
2. **Transmissions from internal nodes.** In each *superstep*, each node in X transmits one message if there is any message received but not sent. *Each node can receive at most one upward message from its children*. Therefore, a node can perform an upward transmission only if its parent has not received an upward message in the superstep. Otherwise, it performs a downward transmission. For each superstep, the algorithm performs the following.
   a. **Transmissions from primaries.** Primaries are scheduled *before* secondaries. Transmissions from primaries are scheduled based on the *vertex-coloring* of H_1 and the order to process nodes is the same as in CDA. Recall that a node performs an upward transmission if its parent has not received a message from its sibling in the same superstep. Otherwise, it performs a downward transmission. Transmissions from primaries require at most 12 time slots.
   b. **Upward transmissions from secondaries.** Secondary nodes are considered in the same order as the broadcast tree is constructed, and a node can perform an upward transmission if its parent has not received a message from its sibling in the same superstep. Upward transmissions from secondaries require at most 16 time slots as shown below.
   c. **Downward transmissions from secondaries.** Once all upward transmissions are scheduled, nodes which are not scheduled for upward transmissions are considered in the same order as the broadcast tree is constructed, and downward transmissions are scheduled. Downward transmissions require at most five time slots.

Below we prove that this algorithm yields a 34-approximation. The following fact will be useful for the analysis of 34-approximation.

Fact 5.4. *In each superstep, each node transmits at most one message, either upward or downward, and receives at most two messages, one from one of its children and one from its parent.*

Note that some of those messages that a node received may be redundant because for example, a node v may receive $m(v)$ when the parent of node v transmits $m(v)$ to its parent.

Lemma 5.5. *All terminal nodes transmit their messages to their parents one by one, which takes at most $|Y|$ time slots.*

Optimal solution requires at least n ($\geq |Y|$) time slots. Therefore transmissions from terminal nodes increase the approximation factor by one.

Lemma 5.6. *The transmissions from primaries require at most 12 time slots in each superstep.*

Lemma 5.7. *The upward transmissions from secondaries require at most 16 time slots.*

Lemma 5.8. *The downward transmissions from secondaries require at most five time slots.*

By Lemmas 5.5-5.8, the algorithm gives a 34-approximation. The proofs of 5.6 and 5.8 are straightforward by vertex coloring properties [5]. Below, we prove Lemma 5.7.

Proof of Lemma 5.7. When a node x sends an upward transmission to its parent $p(x)$ and children $C(x)$, there cannot be any collision at a node $v \in C(x)$ as nodes are considered in the same order as we construct the broadcast tree. Therefore, the following covers all possible cases where x should avoid transmission.

1. There is a node y which sends a message to $z \in D_p(x) - C(x) - \{p(x)\}$. Suppose that there are $\bar{n}_p(x)$ such nodes. Then $\bar{n}_p(x) \leq 3$ and each of those nodes may receive a message from its parent or child. Therefore, x should avoid at most $2\bar{n}_p(x)$ time slots.
2. There is a node y which sends a message to $p(x)$. In this case, y should be the parent of $p(x)$ and x needs to avoid only 1 time slot (at most).
3. There is a node $y \in D(p(x))$ sending a message (but not to $p(x)$), thus creating collision at $p(x)$. This requires x to avoid at most $(16 - \bar{n}_p(x)) - \lceil (16 - \bar{n}_p(x))/3 \rceil$ time slots.

The first and second cases are straightforward. Proposition 5.9 provides the proof for the third case. The maximum number of time slots to avoid in total is 15 when $\bar{n}_p(x) = 3$. Thus, we have proved Lemma 5.7. □

Proposition 5.9. *Suppose that a node x performs an upward transmission and there is a node $y \in D(p(x))$ sending a message (but not to $p(x)$), thus creating collision at $p(x)$. This case requires x to avoid at most $(16 - \bar{n}_p(x)) - \lceil (16 - \bar{n}_p(x))/3 \rceil$ time slots.*

Proof. When a node $y \in D(p(x))$ performs an upward transmission, there is a unique primary parent $p(y)$. To count the number of time slots that a node x needs to avoid in the third case, we count the number of such primary

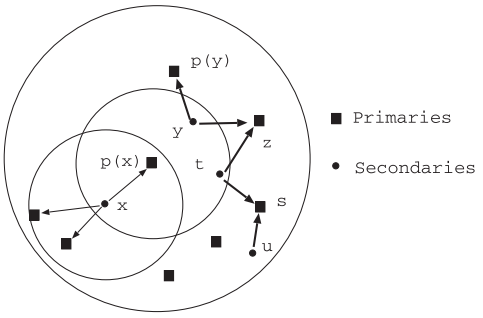

Fig. 3. Proof of proposition 5.9. At least one-third of nodes in $D''(x) = D_p(p(x), 2) - D_p(x) - p(p(p(x)))$ do not interfere with the upward transmission from x.

parents. We can exclude the primaries for the first and second cases, and define $D''(x) = D_p(p(x), 2) - D_p(x) - p(p(p(x)))$. Then $|D''(x)| \leq 19 - (2 + \bar{n}_p(x)) - 1 = 16 - \bar{n}_p(x)$. In the following, we will show that approximately one-third of upward transmissions to them do not create collision with transmission from x (see Fig. 3). This can be shown by proving that for every two primary nodes, there is at least one primary node to which the upward transmission cannot create a collision with upward transmission from x.

Let us consider a node $y \in D(p(x))$ performing an upward transmission. y has a unique primary parent $p(y)$ and at least one primary child z (as y is not in Y). Suppose that an upward transmission to z does not interfere with the transmission from x. Then a primary node with no collision is found and we are done. Suppose now that there is another node t in $D(p(x))$, which is a child of z, sending an upward message to z, thus interfering with the transmission from x as shown in Fig. 3. Using the same argument, t has its primary child s in $D_p(p(x), 2)$. Note that the child of s cannot be in $D(p(x))$ interfering with the transmission from x due to the nature of BFS and the way to construct the broadcast tree. Therefore, for any two primaries in $D''(x)$ ($p(y)$ and z in this example), there is at least one primary (node s) such that the upward transmission to the node does not interfere with the transmission from x.

Therefore, at least $\lceil D''(x)/3 \rceil = \lceil (16 - \bar{n}_p(x))/3 \rceil$ primaries are not causing collisions with transmissions from x and therefore, we have the proposition. □

This algorithm yields a 34-approximation. Even though the theoretical bound of ICDA is weaker than that of CDA, the experimental results (Section 6) show that it provides comparable performance as CDA. In fact, for large networks (300 nodes or more), ICDA performs better than CDA.

Theorem 5.10. *ICDA gives an approximation factor of 34.*

Proof. By Lemmas 5.5-5.8, the theorem follows. □

6 Experimental Evaluation

6.1 Simulation Setup

We place wireless nodes in a square (e.g., $1{,}000\ \mathrm{m} \times 1{,}000\ \mathrm{m}$) uniformly at random, while varying the number of nodes and the size of square. We use a fixed transmission range of 200 m and assume two nodes can communicate if they are within the transmission range of each other. For one-to-all broadcast experiments, we select a source uniformly at random, and the source becomes the root of broadcast tree. For all-to-all broadcast, every node is a source and sends one message each. We construct a broadcast tree for each node as the root and choose the one with minimum height. For each scenario, we use 20 runs with different random seeds. We focus on two performance metrics when we report our experiment results. First, we consider the **latency metric** to complete the broadcast. We display the average as well as the minimum and maximum using error bars. We also use the **number of transmissions** as another metric and report the average.

In our experiments, we compare our proposed algorithms with existing schemes. Gandhi et al. [4] present a one-to-all algorithm (which we call GPM) and an all-to-all algorithm (which we call MSB). In MSB, each node creates its own broadcast tree and schedules transmissions greedily. We also compare our proposed algorithms with schemes by Huang et al. [5], [6]. We consider two of their one-to-all algorithms [5]: Enhanced Broadcast Scheduling (EBS) and Pipelined Broadcast Scheduling (PBS). Similarly to GPM, EBS handles one level in a BFS tree at a time. PBS first sends a message to all nodes in a maximal independent set (MIS) along a shortest-path tree. In the second phase, PBS schedules the nodes in the MIS to transmit to their neighbors. Huang et al. [6] also propose an all-to-all algorithm called Interleaved Gossiping Algorithm (IGA), which works similar to CDA except for the transmission schedule of secondary nodes. In IGA, secondary nodes are divided by three sets depending on their BFS level. Then, each set is divided into four noninterfering groups by running Iterative Minimal Covering algorithm [5], which results in the bound of 12 time slots for secondary nodes.

Optimization Heuristics. In our experiments, we use optimization heuristics that do not affect the worst-case bound, while significantly improving practical performance. Specifically, we optimize PBS such that a node in MIS does not transmit if all its neighbors have received the message from a transmission in the first phase. Also, although the original description of both EBS and PBS uses G^2 to find independent sets that correspond to units of simultaneous transmissions (e.g., in the second phase of PBS), we can significantly improve the performance by using a carefully chosen subgraph when finding the independent sets. In the original description of EBS, at each level, secondary nodes are scheduled strictly after primary nodes. We also optimize EBS by allowing secondary nodes to transmit opportunistically at the same slots as primary nodes if they do not interfere with the primary nodes. In IGA, while the worst case bound for each transmission phase is 24 time slots, we observe many idle time slots in our experiments. In our experiments, we remove those unnecessary time slots to improve the performance of their algorithm.

We also use two optimization heuristics for CDA, ICDA, and IGA. First, we allow nodes to transmit messages opportunistically after sending a message within a superstep if it does not affect the worst-case bound. This strategy helps eventually reduce the number of rounds to finish the broadcast. Second, we allow a node to receive messages from nontree neighbors if it hears a message collision free. This can potentially eliminate some of transmissions from its tree neighbors to the node, decreasing the total broadcast time.

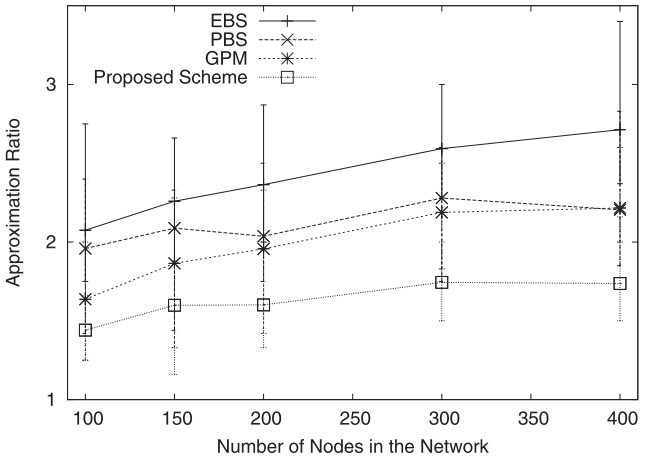

Fig. 4. Average approximation ratio of one-to-all broadcast. We use a fixed-size square of $1{,}000\ \mathrm{m} \times 1{,}000\ \mathrm{m}$).

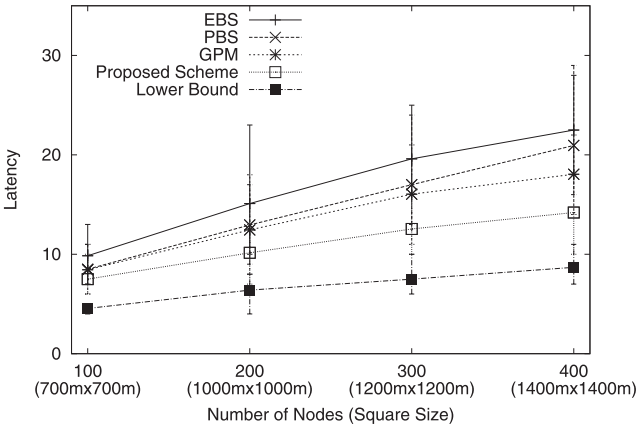

Fig. 5. Average latency of one-to-all broadcast. We vary the square size to keep the node density similar.

6.2 Results for One-to-All Broadcast

While we have experimented with various settings, we only report a set of representative results in the rest of this section. In Fig. 4, we present the average approximation ratios when we vary the number of nodes within a fixed-size square $(1{,}000\ \mathrm{m} \times 1{,}000\ \mathrm{m})$. Our proposed algorithm consistently outperforms existing schemes by 12-40 percent. Specifically, in the 400-node scenario, the average approximation ratio of our algorithm is 1.74, which is around 21 percent smaller than that of GPM (2.22) and PBS (2.21) and 40 percent smaller than that of EBS (2.92). Note that for each level in a BFS tree, EBS separates MIS nodes and non-MIS nodes (that are parents of next-level MIS nodes), which increases the completion time. In contrast, in our scheme, primary nodes and secondary nodes at the same level can be scheduled at a same time slot. Combined with the two-phase procedure for handling one level, our scheme significantly improves broadcast latency. PBS performs similar to GPM, and its asymptotic ratio does not directly translate to good performance in practice. As the network becomes denser with more nodes, the approximation ratio of our scheme goes up slightly, but the performance improvement over existing schemes is similar or larger. Although our analytical bound is 12, we observe that the performance of our scheme in our simulations is much closer to optimal and stays similar when we vary the node density.

In Table 1, we report the average height of breadth-first search tree rooted at the source (which is used as lower bound) for the experiments reported in Fig. 4. For instance, in the 400-node case, the average lower bound is 6.25. On the other hand, the average latency value of our proposed scheme is 10.85, which results in the approximation ratio of 1.74 as shown in Fig. 4. We also identify the maximum, average, and minimum degrees for each run and report the average of each value in the table. Even though nodes are distributed uniformly at random, we observe that the difference between average and maximum degrees is quite high (e.g., 21 versus 64 neighbors in 400-node networks on average). This is because nodes close to the boundary of the square have much fewer neighbors, causing the average degrees to go down. We note that the 100-node scenario corresponds to fairly sparse networks—around 10 percent of random instances resulted in partitioned networks, which are not part of 20 runs used. (With 50 nodes, only 20 percent of instances were connected.)

TABLE 1
Average BFS Tree Height and Node Degrees

no. nodes	avg. BFS tree height	avg. degrees max	avg	min
100	6.85	19.35	5.32	2.35
150	6.6	27.20	7.89	3.90
200	6.4	34.55	10.45	5.15
300	6.2	49.25	15.73	8.45
400	6.25	64.35	20.98	11.85

In Fig. 5, we present the average values of actual latency for broadcast algorithms as well as the height of BFS tree (i.e., lower bound). In this set of experiments, we vary both the number of nodes and the square size, so that the average number of neighbors is maintained similar. We observe that our proposed algorithm consistently outperforms existing schemes by 11-37 percent. Specifically, in the 400-node scenario, the average latency of our algorithm is 14.2, while the lower bound is 8.7 (i.e., approximation ratio is around 1.64). This is around 21 percent better than GPM (18.1), 32 percent better than PBS (21.0), and 37 percent better than EBS (22.5). Although our optimization scheme for EBS significantly improves the performance over the original description, the separation between levels and between MIS and non-MIS nodes leads to longer latency. In contrast, in our scheme, primary nodes and secondary nodes at the same level can be scheduled at a same time slot. PBS performs similar to GPM, and its asymptotic ratio does not directly translate to good performance in practice. While the BFS tree height increases as the square size increases, our proposed scheme maintains the approximation ratio around 1.65. Although our analytical bound is 12, we observe that the performance of our scheme in practice is much closer to optimal.

In the previous experiments, the interference range was the same as the transmission range. In practice, however, it is possible that the interference range is larger than the transmission range. We perform experiments by varying the interference range and accordingly adjusting the set of interfering nodes as described in Section 4. In Fig. 6, we present our results when we, respectively, place 200 nodes and 400 nodes in a $1{,}000\ \mathrm{m} \times 1{,}000\ \mathrm{m}$ area. We observe that the broadcast latency increases as the interference ranges

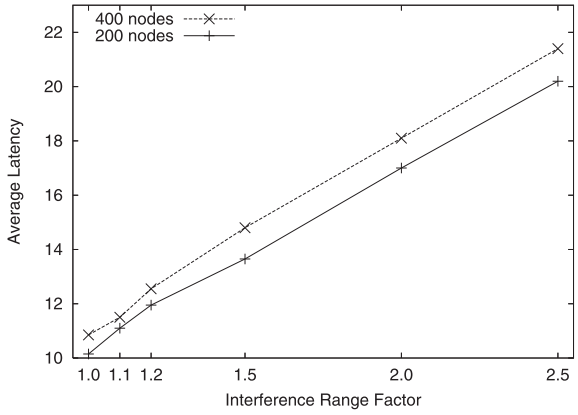

Fig. 6. Latency change when we vary the interference range.

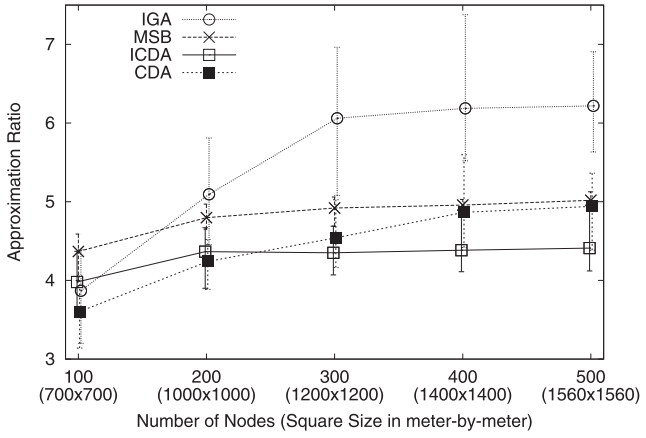

Fig. 7. Average approximation ratio of all-to-all broadcast. Node count and square size vary.

increases. This is expected, because as a transmission interferes with more nodes, nodes potentially need to wait longer before they can transmit their message without causing collisions.

Packet Transmissions. Our results show that despite some nodes possibly transmitting up to twice (Section 4), our one-to-all broadcast algorithm leads to similar average number of transmissions to that of GPM (less than 6 percent difference in all cases). The number of transmissions by EBS is similar to our scheme, and PBS sends 13-22 percent more than GPM in the experiments shown in Fig. 5. Specifically, when there are 200 nodes in $1{,}000\ \mathrm{m} \times 1{,}000\ \mathrm{m}$) square, GPM on average transmits 32.8 times, our scheme 33.2, EBS 34.0, and PBS 37.3. In this example, among the 33.2 transmissions for our scheme, on average 27.2 nodes transmit once, and only 3.0 nodes transmit twice. In PBS, 2.8 out of around 34.5 transmitting nodes transmit twice on average in this scenario. In all our experiments, we observe that our scheme requires 7-14 percent of relay nodes to transmit twice, which is similar to 5-14 percent in the case of PBS.

6.3 Results for All-to-All Broadcast

In Fig. 7, we present the average approximation ratio of our all-to-all broadcast schemes (CDA and ICDA), MSB and IGA when we vary the number of nodes and the square size. We first observe that the performance of CDA and ICDA in practice is much better than the analytical bound (20 and 34). We also observe that in all schemes, the ratio increases as the network becomes larger. CDA performs well when the network size is small (e.g., around 17 percent better than MSB in the 100-node case). However, the performance difference between CDA and MSB becomes smaller in larger networks. This is because both schemes first send all packet to the root before distributing them, and the initial latency increases with the network size growth. Except for the 100-node case, we observe that our all-to-all schemes achieve significant performance improvement over IGA. Specifically, CDA consistently outperforms IGA by 20-33 percent. Even though IGA works very similar to CDA, IGA uses three times more time slots for secondary nodes than CDA and thus takes longer time to complete all-to-all broadcasting.

Although ICDA does not perform as well as CDA for a small network size, ICDA consistently outperforms MSB by 7-11 percent in all cases. In fact, the performance gap between ICDA and MSB becomes larger as the network size grows, which indicates the benefit of interleaved transmissions. The performance gap between ICDA and IGA grows as large as 41 percent as the network size grows. Specifically, with 200 nodes in $1{,}000\ \mathrm{m} \times 1{,}000\ \mathrm{m}$) square, CDA takes around 850 (or approximation ratio of 4.24) and ICDA takes around 870 time units (or ratio of 4.36) to complete the broadcast of 200 messages, while MSB takes around 960 time units (or ratio of 4.80), and IGA takes around 1,074 time units (or ratio of 5.37).

We also observe that our optimization heuristics are effective in reducing the number of rounds to complete the broadcast. For instance, when ICDA does not use the optimization schemes described in the previous section, it takes around 1,200 time units to finish. This indicates that while our basic all-to-all algorithms give a constant-factor guarantee on the number of transmissions, our optimization heuristics can further reduce the broadcast latency.

In terms of transmission overhead, our results show that both CDA and ICDA consistently use fewer transmissions than MSB (by 13-19 percent depending on the scenarios), while CDA and ICDA perform similarly. Due to the similarity in their scheduling mechanism, the transmission overhead of IGA and CDA is almost the same, although IGA uses slightly more transmissions (e.g., by 1-2 percent) in our experiments.

7 Conclusions

In this paper, we presented approximation algorithms for broadcasting in multihop wireless networks. Our algorithm for ONE-TO-ALL BROADCASTING gives a 12-approximate solution, and the algorithms for ALL-TO-ALL BROADCASTING give approximation ratios of 20 and 34. Our simulation results show that in practice, these proposed schemes perform much better than the theoretical bound and achieve up to 37 percent latency performance improvement over existing schemes.

Acknowledgments

A preliminary version of this paper appeared in the Proceedings of the 28th IEEE INFOCOM, 2009. This work was supported by US National Science Foundation Awards CCF-0830569 and CCF-1048606. This work was supported in part by the National Research Foundation of Korea Grant funded by the Korean Government (NRF-2012R1A6A3A01012020).

REFERENCES

[1] C. Ho, K. Obraczka, G. Tsudik, and K. Viswanath, "Flooding for Reliable Multicast in Multi-Hop Ad Hoc Networks," *Proc. Third Int'l Workshop Discrete Algorithms and Methods for Mobile Computing and Comm.,* pp. 64-71, 1999.

[2] J. Jetcheva, Y. Hu, D. Maltz, and D. Johnson, "A Simple Protocol for Multicast and Broadcast in Mobile Ad Hoc Networks," IETF Internet draft, July 2001.

[3] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The Broadcast Storm Problem in a Mobile Ad Hoc Network," *Proc. ACM/IEEE MobiCom,* pp. 151-162, 1999.

[4] R. Gandhi, A. Mishra, and S. Parthasarathy, "Minimizing Broadcast Latency and Redundancy in Ad Hoc Networks," *IEEE/ACM Trans. Networking,* vol. 16, no. 4, pp. 840-851, Aug. 2008.

[5] S. Huang, P. Wan, X. Jia, H. Du, and W. Shang, "Minimum-Latency Broadcast Scheduling in Wireless Ad Hoc Networks," *Proc. IEEE INFOCOM,* pp. 733-739, 2007.

[6] S.C.-H. Huang, H. Du, and E.-K. Park, "Minimum-Latency Gossiping in Multi-Hop Wireless Networks," *Proc. ACM MobiHoc,* pp. 323-330, 2008.

[7] H. Lim and C. Kim, "Multicast Tree Construction and Flooding in Wireless Ad Hoc Networks," *Proc. Third ACM Int'l Workshop Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM '00),* pp. 61-68, 2000.

[8] W. Peng and X.-C. Lu, "On the Reduction of Broadcast Redundancy in Mobile Adhoc Networks," *Proc. ACM MobiHoc,* Aug. 2000.

[9] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint Relaying: An Efficient Technique for Flooding in Mobile Wireless Networks," Technical Report RR-3898, INRIA, Feb. 2000.

[10] W. Peng and X. Lu, "AHBP: An Efficient Broadcast Protocol for Mobile Adhoc Networks," *J. Science and Technology,* vol. 16, pp. 114-125, 2000.

[11] J. Sucec and I. Marsic, "An Efficient Distributed Network-Wide Broadcast Algorithm for Mobile Adhoc Networks," technical report, Rutgers Univ., 2000.

[12] W. Peng and X. Lu, "Efficient Broadcast in Mobile Ad Hoc Networks Using Connected Dominating Sets," *J. Software,* 1999.

[13] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating Sets and Neighbor Elimination-Based Broadcasting Algorithms in Wireless Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 13, no. 1, pp. 14-25, Jan. 2002.

[14] L. Orecchia, A. Panconesi, C. Petrioli, and A. Vitaletti, "Localized Techniques for Broadcasting in Wireless Sensor Networks," *Proc. Joint Workshop Foundations of Mobile Computing (DIALM-POMC '04),* pp. 41-51, 2004.

[15] H.F. Salama, D.S. Reeves, and Y. Viniotis, "The Delay-Constrained Minimum Spanning Tree Problem," *Proc. IEEE Second Symp. Computers and Comm.,* pp. 699-704, 1997.

[16] I. Chlamtac and S. Kutten, "On Broadcasting in Radio Networks—Problem Analysis and Protocol Design," *IEEE Trans. Comm.,* vol. 33, no. 12, pp. 1240-1246, Dec. 1985.

[17] I. Chlamtac and O. Weinstein, "The Wave Expansion Approach to Broadcasting in Multihop Radio Networks," *IEEE Trans. Comm.,* vol. 39, no. 3, pp. 426-433, Mar. 1991.

[18] S. Basagni, I. Chlamtac, and D. Bruschi, "A Mobility-Transparent Deterministic Broadcast Mechanism for Ad Hoc Networks," *IEEE/ACM Trans. Networking,* vol. 7, no. 6, pp. 799-807, Dec. 1999.

[19] Z. Chen, C. Qiao, J. Xu, and T. Lee, "A Constant Approximation Algorithm for Interference Aware Broadcast in Wireless Networks," *Proc. IEEE INFOCOM,* pp. 740-748, 2007.

[20] R. Tiwari, T.N. Dinh, and M.T. Thai, "On Approximation Algorithms for Interference-Aware Broadcast Scheduling in 2D and 3D Wireless Sensor Networks," *Proc. Fourth Int'l Conf. Wireless Algorithms, Systems, and Applications (WASA '09),* pp. 438-448, 2009.

[21] R. Mahjourian, F. Chen, R. Tiwari, M. Thai, H. Zhai, and Y. Fang, "An Approximation Algorithm for Conflict-Aware Broadcast Scheduling in Wireless Ad Hoc Networks," *Proc. ACM MobiHoc,* pp. 331-340, 2008.

[22] P.-K. Hung, J.-P. Sheu, and C.-S. Hsu, "Scheduling of Broadcasts in Multihop Wireless Networks," *Proc. European Wireless,* 2002.

[23] B. Williams and T. Camp, "Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks," *Proc. ACM MobiHoc,* pp. 194-205, 2002.

[24] P. Bateman and P. Erdös, "Geometrical Extrema Suggested by a Lemma of Besicovitch," *The Am. Math. Monthly,* vol. 58, pp. 306-314, May 1951.

[25] C. Florens and R. McEliece, "Packets Distribution Algorithms for Sensor Networks," *Proc. IEEE INFOCOM,* pp. 1063-1072, 200.

Rajiv Gandhi received the BE degree in electrical engineering from Bombay University (Victoria Jubilee Technological Institute), the MS degree in computer science from Virginia Tech, and the PhD degree in computer science in 2003 from the University of Maryland, College Park. Before starting the PhD degree, he worked as a software engineer at Qualcomm, Inc. He is an associate professor of computer science at Rutgers University, Camden. His research interests include approximation algorithms for NP-hard problems arising in the domain of scheduling, wireless networks, and clustering.

Yoo-Ah Kim received the PhD degree in computer science from the University of Maryland in 2005. She is a research fellow in the National Center for Biotechnology Information (NCBI) at the National Institutes of Health. Her research interests include design and analysis of algorithms, bioinformatics, and computer networks.

Seungjoon Lee received the bachelor's and master's degrees in computer science from Seoul National University, Korea, in 1996 and 2000. He received the PhD degree in computer science from University of Maryland, College Park in 2006. His research interests include network management, content distribution, and mobile computing. He is a senior member of technical staff at AT&T Research, Florham Park, New Jersey.

Jiho Ryu received the PhD degree from the School of Computer Science and Engineering, Seoul National University, Korea, in 2012. He joined the Ulsan National Institute of Science and Technology (UNIST), Korea, as a postdoctoral researcher. His research interests include wireless scheduling, wireless LANs, sensor networks, wireless mesh networks, and real-time systems. He is a member of the IEEE.

Peng-Jun Wan received the BS degree is applied mathematics from Tsinghua University in 1990, the MS degree in operations research and control theory from the Chinese Academy of Sciences in 1993, and the PhD degree in computer science from the University of Minnesota in 1997. He is currently a professor of Computer Science and Engineering at the Illinois Institute of Technology. His research interests include design and analysis of approximation algorithms for optimization problems in wireless networks and optical networks.