

# Minimizing Broadcast Latency and Redundancy in Ad Hoc Networks

Rajiv Gandhi Arunesh Mishra Srinivasan Parthasarathy

**Abstract**—Network wide broadcasting is a fundamental operation in ad hoc networks. In broadcasting, a source node sends a message to all the other nodes in the network. In this paper, we consider the problem of collision-free broadcasting in ad hoc networks. Our objective is to minimize the latency and the number of transmissions in the broadcast. We show that minimum latency broadcasting is NP-Complete for ad hoc networks. We also present a simple distributed collision-free broadcasting algorithm for broadcasting a message. For networks with bounded node transmission ranges, our algorithm *simultaneously* guarantees that the latency and the number of transmissions are within  $O(1)$  times their respective optimal values. Our algorithm and analysis extend to the case when multiple messages are broadcast from multiple sources. Experimental studies indicate that our algorithms perform much better in practice than the analytical guarantees provided for the worst case.

## I. INTRODUCTION

Ad hoc networks are a set of wireless mobile nodes which communicate with one another. Nodes in these networks do not rely on any pre-existing routing infrastructure for communication, but instead communicate either directly or with the help of other intermediate nodes in the network. The distributed, wireless and self-configuring nature of ad hoc networks make them suitable for a wide variety of applications. At the same time these characteristics also introduce several challenging and interesting research issues in the design of communication protocols for these networks.

Network wide broadcasting is a fundamental operation in ad hoc networks. Its goal is to transmit a message from a source to all the other nodes in the network. Several ad hoc network protocols assume the availability of an underlying broadcast service. Applications which make use of broadcasting include LAN Emulation, host paging, sending an alarm, and establishing unicast routes [1]. Broadcasting is also a common operation in many distributed computing applications and can be used for service or resource discovery in unstructured environments. In the absence of other mechanisms, broadcast

also serves as a last resort for other group communication operations such as multicast.

Any communication protocol for ad hoc networks should contend with the issue of interference in the wireless medium. 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 a collision has occurred at the common node. In multi-hop ad hoc networks where all the nodes may not be within the transmission range of the source, intermediate nodes may need to assist in the broadcast operation by transmitting the message to other remote nodes in the network. Transmissions use up valuable resources in the network such as power and bandwidth. Hence, it is important to choose the intermediate nodes carefully so as to avoid redundancy in transmissions. Another metric of interest in broadcast protocols is the end-to-end broadcast latency which is the time taken by the message to reach all the nodes in the network.

One of the earliest broadcast mechanisms proposed in the literature is flooding [2], [3], where every node in the network transmits a message to its neighbors after receiving it. Although flooding is extremely simple and easy to implement, it was shown in [1] 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 lot of research has been directed towards designing broadcast protocols which are collision-free and which reduce redundancy by reducing the number of transmissions.

Recently, there has been a tremendous interest for supporting real-time multimedia traffic over ad hoc networks. Ad hoc sensor networks are also expected to play a big role in military communications, disaster relief and rescue operations. These applications impose stringent end-to-end latency requirements on the underlying protocols. In light of these real-time applications, it is necessary to have broadcast protocols that can meet the combined requirements of collision-free delivery, low end-to-end latency and low redundancy. None of the existing broadcast protocols address all these issues together.

In this paper, we present a rigorous treatment of collision-free broadcasting in ad hoc networks. Our contributions are summarized below.

### A. Our Contributions

- We show that minimum latency broadcasting in ad hoc networks is NP-Complete (NPC).
- For networks with bounded node transmission ranges, we present a collision-free broadcast algorithm which

A preliminary version of this paper has appeared in MobiHoc'03.

Rajiv Gandhi is with the Department of Computer Science, Rutgers University, Camden, NJ 08102. Part of this work was done when the author was at the University of Maryland, College Park, and was supported by NSF Award CCR-9820965. Part of this work is supported by Rutgers Research Council Grant and the Lindback Minority Junior Faculty Award. E-mail: rajivg@camden.rutgers.edu.

Arunesh Mishra is with the Computer Sciences Department, University of Wisconsin-Madison, WI 53706-1481. E-mail: arunesh@cs.wisc.edu.

Srinivasan Parthasarathy is with IBM T. J. Watson Research Center, Hawthorne, NY, 10532. Work done while at the Department of Computer Science, University of Maryland, College Park, MD 20742. Research supported by NSF Award CCR-0208005 and NSF ITR Award CNS-0426683. E-mail: spartha@us.ibm.com.

*simultaneously* produces provably good solutions in terms of latency and the number of transmissions. Specifically, given an ad hoc network and a collection of broadcast messages, let  $OPT_{lat}$  denote the optimal latency achievable by *any* broadcast algorithm; independently, let  $OPT_{ret}$  denote the optimal number of transmissions achievable by *any* broadcast algorithm. We present a single broadcast algorithm whose latency and number of transmissions are simultaneously guaranteed to be  $O(1)$  times  $OPT_{lat}$  and  $OPT_{ret}$ , respectively.

- We extend our results to the case when multiple messages are broadcast from multiple sources, in networks with uniform node transmission ranges. We study the performance of our algorithms under various network conditions through simulations.

## II. RELATED WORK

Flooding is one of the earliest protocols for multicasting and broadcasting in ad hoc networks [2], [3]. In flooding, every node in the network transmits the message to its neighbors after receiving it. Flooding can lead to severe contention, collision and redundant transmissions: a situation referred to as broadcast storm [1].

In a series of papers [4], [5], [6], it was proposed that a connected dominating set (*CDS*) can be used as a virtual backbone for routing in ad hoc networks. A dominating set in a graph  $G = (V, E)$  is defined as a set of vertices  $V' \subseteq V$ , such that every node in  $V \setminus V'$  is adjacent to some node in  $V'$ . A connected dominating set (*CDS*) is a dominating set whose induced subgraph is connected. An *MCDS* is a *CDS* of minimum size. The notion of *CDS* occurs naturally in broadcast protocols since the set of transmitting nodes in any broadcast protocol forms a *CDS*. Minimizing the number of transmissions is *equivalent* to finding an *MCDS* in a graph.

*MCDS* is NPC for unit disk graphs (*UDGs*) [7], [8]. Our work shows that minimizing broadcast latency in ad hoc networks is NPC irrespective of the number of transmissions. *UDGs* model ad hoc networks with uniform node transmission ranges. Several approximation algorithms for *MCDS* in *UDGs* are known [9], [10], [11]. Although these results lead to efficient broadcast algorithms in terms of the number of transmissions, they do not directly yield low broadcast latency.

Several techniques have been proposed where nodes make use of their neighborhood information to determine if they need to transmit a message [12], [13], [14], [15], [16], [17], [18], [19] in order to reduce broadcast redundancy and contentions. The main point of departure of our work from the above approaches is that we analyze the computational complexity of efficient broadcasting, and also present reliable, and collision-free algorithms with constant-factor approximation guarantees in terms of both broadcast latency and redundancy.

There has been relatively less research directed towards low latency broadcasting in ad hoc networks. There has been some work on latency-constrained broadcasting in wired networks [20] and some results do exist for radio networks whose models are essentially the same as ours. In particular, Chlamtac and Kutten [21] show that minimum latency broadcast

scheduling is NPC for general (non-geometric) graphs. This result does not directly extend to ad hoc networks which are modeled by a restricted class of geometric graphs called disk graphs. In [22], Chlamtac and Kutten gave an algorithm for constructing a broadcast tree and for collision-free scheduling along this broadcast tree. 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 graph theoretic radius of the network.

Basagni *et al.* [23] present a mobility transparent broadcast scheme for mobile multi-hop 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 extremely 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 broadcast schedule can be computed efficiently. In addition, they do not bound the number of transmissions. Other results which deal with broadcasting in unknown topologies include [24], [25].

Hung *et al.* [26] provide centralized and distributed algorithms for broadcasting and experimentally 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 [27] 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.

Finally, some work is done in optimizing the broadcast time in radio networks which is the actual time required by a broadcast message to reach all the network nodes under the same interference model as that of ours. However, the computation of a broadcast schedule before a message originates is disallowed. All results [28], [29], [30], [31] obtained for this problem are for arbitrary graphs.

## III. PRELIMINARIES

### A. Network Model

We model an ad hoc network using a directed graph  $G = (V, E)$ . The nodes in  $V$  are embedded in the plane. Each node  $u \in V$  has a transmission range,  $range(u) \in [r_{min}, r_{max}]$ . Let  $d(u, v)$  denote the Euclidean distance between  $u$  and  $v$ . An arc  $(u, v) \in E$  iff  $v$  is in the transmission range of  $u$ , i.e.,  $d(u, v) \leq range(u)$ . Such graphs are called Disk Graphs. When the node transmission ranges are uniform,  $G$  is called a Unit Disk Graph (UDG). If  $(u, v) \in E$  and  $(v, u) \notin E$  then we say that  $(u, v)$  is uni-directional, otherwise  $(u, v)$  is bi-directional. A UDG can be treated as an undirected graph since all its edges are bi-directional.

We assume that time is discrete. Since the medium of transmission is wireless, whenever a node transmits a message,

all its out-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. A node  $w$  receives a message collision-free iff  $w$  hears the message without any collision.

We note that the disk graph transmission model and our interference assumptions are approximations of reality. However, efficient broadcasting is already NPC under these assumptions; while our guarantees are derived under these simplified assumptions, we believe our work lays the theoretical foundations for algorithms in the future under more realistic network models.

### B. Problem Statement

We are given a disk graph  $G = (V, E)$  and a set of messages  $M = \{1, 2, \dots, m\}$ . 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 broadcast *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 broadcast schedule in which the latency and the number of transmissions are minimized.

### C. Background: Approximation Algorithms

A common approach to NPC problems is to develop *approximation algorithms*, which run in polynomial time and which produce provably good solutions.

**Definition.** An  $\alpha$ -*approximation* algorithm for a minimization problem  $\Pi$  is a poly-time algorithm such that for every instance  $I$  of  $\Pi$  it produces a solution of cost at most  $\alpha \text{OPT}(I)$ ,  $\alpha > 1$ , where  $\text{OPT}(I)$  refers to cost of an optimal solution for instance  $I$ . In this case, we also say that the approximation algorithm has an approximation factor of  $\alpha$ .

## IV. NP-COMPLETENESS

We now show that the problem of minimum latency broadcasting in ad hoc networks is NPC even in the case when a single source needs to broadcast a single message. The decision version of our problem is as follows.

**Input:** A set of nodes *embedded* in a plane and a source node  $s$  that wants to broadcast a single message. The transmission range of each node  $u$  is  $range(u)$ . Arc  $(u, v)$  exists iff  $d(u, v) \leq range(u)$ .

**Question:** Is there a broadcast schedule with latency  $\leq L$ ?

We prove that this problem is NPC via a reduction from 3-SAT. We define 3-SAT below.

**Input:** Set of variables,  $X = \{x_1, x_2, \dots, x_n\}$  and a collection of clauses,  $C = \{c_1, c_2, \dots, c_m\}$  over  $X \cup \bar{X}$  such that the

size of each clause in  $C$  is exactly three.

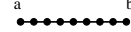
**Question:** Is there a truth assignment that satisfies  $C$ ?

### A. Reduction

Let  $I$  be an instance of 3-SAT. We will transform  $I$  into an instance  $\rho(I)$  of the minimum latency broadcast problem. Specifically, we will show that  $I$  has a satisfying assignment iff  $\rho(I)$  has a schedule of latency  $L$ , where  $L = 32 \cdot (m+n+3)$ , where  $m$  and  $n$  are the number of clauses and variables in  $I$  respectively. We now describe the reduction  $\rho$ .

For any node  $u$  in  $\rho(I)$ , define its *characteristic vector*  $cv(u)$  to be  $(u.x, u.y, range(u))$ , where  $(u.x, u.y)$  represents the co-ordinates of node  $u$ . Let  $D(u, v)$  denote the shortest path distance between  $u$  and  $v$ , i.e., the number of arcs on the shortest path from  $u$  to  $v$ . Our reduction is a combination of several gadgets which we describe below.

**Filler gadget:** Let  $a$  and  $b$  denote two nodes such that either  $a.x = b.x$  or  $a.y = b.y$ , i.e., the line segment  $(a, b)$  is either vertical or horizontal. Let  $\delta$  be such that  $d(a, b)/\delta$  is a positive integer.  $Filler(a, b, \delta)$  places a set of  $k$  equi-spaced nodes along the line segment  $(a, b)$ , where  $k = \frac{d(a, b)}{\delta} - 1$ . The range of each node in this set is  $\delta$ . Thus, let  $a.y = b.y$  and  $a.x < b.x$ ; if we let  $a_0 = a$  and  $a_{k+1} = b$  then for all  $i \in \{1, 2, \dots, k\}$ ,  $cv(i) = (a_{i-1}.x + \delta, a_{i-1}.y, \delta)$ . This is illustrated in Figure 1.

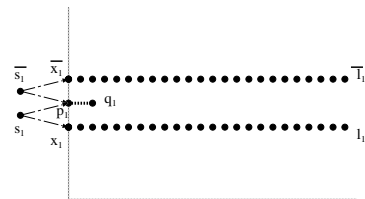


**Figure 1.**  $Filler(a, b, \delta)$ : the distance between successive points is  $\delta$ .

**Variable gadget:** We now describe the gadget corresponding to each variable  $x_i \in X$ . For notational convenience, we also use  $x_i$  and  $\bar{x}_i$  to denote nodes in the gadget corresponding to variable  $x_i$ . We also use  $X$  to denote the set  $\{x_i, \bar{x}_i | 1 \leq i \leq n\}$ . The references will be clear from the context.

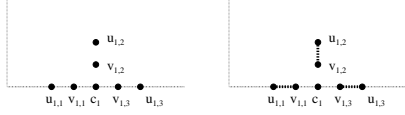
Corresponding to each variable  $x_i \in X$ , we create the nodes in  $\{x_i, \bar{x}_i, s_i, \bar{s}_i, p_i, q_i, l_i, \bar{l}_i\}$  whose characteristic vectors are: (i)  $cv(x_i) = (0, 32i + 16, 1)$ ; (ii)  $cv(\bar{x}_i) = (0, 32i + 32, 1)$ ; (iii)  $cv(s_i) = (-4, 32i + 20, d(s_i, p_i))$  (iv)  $cv(\bar{s}_i) = (-4, 32i + 28, d(\bar{s}_i, \bar{p}_i))$ ; (v)  $cv(p_i) = (0, 32i + 24, d(p_i, q_i)/(L-3))$ ; (vi)  $cv(q_i) = (4, 32i + 24, 0)$ ; (vii)  $cv(l_i) = (32(m+1), x_i.y, 0)$ ; (viii)  $cv(\bar{l}_i) = (32(m+1), \bar{x}_i.y, 0)$ .

In addition to the eight nodes just created, the gadget also consists of nodes in  $Filler(x_i, l_i, 1) \cup Filler(\bar{x}_i, \bar{l}_i, 1) \cup Filler(p_i, q_i, d(p_i, q_i)/(L-3))$ . Note that while  $p_i$  is within the range of both  $s_i$  and  $\bar{s}_i$ ,  $x_i$  ( $\bar{x}_i$ ) is only within the range of  $s_i$  ( $\bar{s}_i$ ). This construction is illustrated in Figure 2.



**Figure 2.** A *variable-gadget*.  $p_i$  and  $q_i$  are connected through a *Filler*.  $s_i$  ( $\bar{s}_i$ ) is equidistant from  $x_i$  ( $\bar{x}_i$ ) and  $p_i$ . The range of  $s_i$  includes  $x_i$  and  $p_i$  but not  $\bar{x}_i$ .

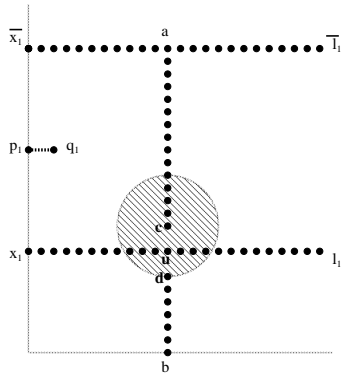
**Clause gadget:** The gadget for each clause  $c_j \in C$  consists of nodes in  $\{c_j, u_{j_1}, u_{j_2}, u_{j_3}, v_{j_1}, v_{j_2}, v_{j_3}\}$  whose characteristic vectors are: (in the expressions that follow,  $range(u_{j_1}), range(u_{j_2})$  and  $range(u_{j_3})$  will be specified later) (i)  $cv(c_j) = (32j, 0, 0)$ ; (ii)  $cv(u_{j_1}) = (c_j.x - 8, 0, range(u_{j_1}))$ ; (iii)  $cv(u_{j_2}) = (c_j.x, 8, range(u_{j_2}))$ ; (iv)  $cv(u_{j_3}) = (c_j.x + 8, 0, range(u_{j_3}))$ ; (v)  $cv(v_{j_1}) = (c_j.x - 4, 0, 4)$ ; (vi)  $cv(v_{j_2}) = (c_j.x, 4, 4)$ ; (vii)  $cv(v_{j_3}) = (c_j.x + 4, 0, 4)$ . This gadget is illustrated in Figure 3.



**Figure 3.** A clause-gadget. The left and right figures indicate the gadget before and after placing the *Filler* nodes respectively.

**VFiller gadget:** The VFiller gadget is used to connect nodes with the same  $x$  coordinates and is illustrated in Figure 4. In this illustration, node  $a$  belongs to  $Filler(\bar{x}_1, \bar{l}_1, 1)$  and is connected to node  $b$  through a VFiller. All nodes in the VFiller gadget have a range of 1 and successive nodes from  $a$  to  $b$  in the gadget are a distance of 1 apart from each other. The exceptions to this rule occur when the VFiller “intersects” with some other HFiller (which does not contain  $a$ ) at a node  $u$ . In this case, we do not include node  $u$  and the nodes immediately above and below node  $u$  in the VFiller. Further, we set the range of node  $c$ , which lies two units above node  $u$ , to 4 units (instead of 1). This increased range enables node  $c$  to transmit to node  $d$  which is two units below node  $u$ .

Formally, let  $H$  be the set of nodes in  $\bigcup_i Filler(x_i, l_i, 1) \cup Filler(\bar{x}_i, \bar{l}_i, 1)$ . These nodes were created during the construction of the gadgets for variables. Let  $a$  and  $b$  be nodes with integer coordinates such that  $a.x = b.x$  and  $a.y > b.y$ . Define  $VFiller(a, b, 1) = Filler(a, b, 1) \setminus R$ , where  $R = \{u | \exists v \in H \text{ such that } (v \neq a) \text{ and } (u.x = v.x) \text{ and } |u.y - v.y| \leq 1\}$ . The transmission range of a node  $u$ ,  $range(u) = 4$  if  $\exists v \in H$  such that  $(u.x = v.x)$  and  $(u.y - v.y = 2)$ ; otherwise,  $range(u) = 1$ .

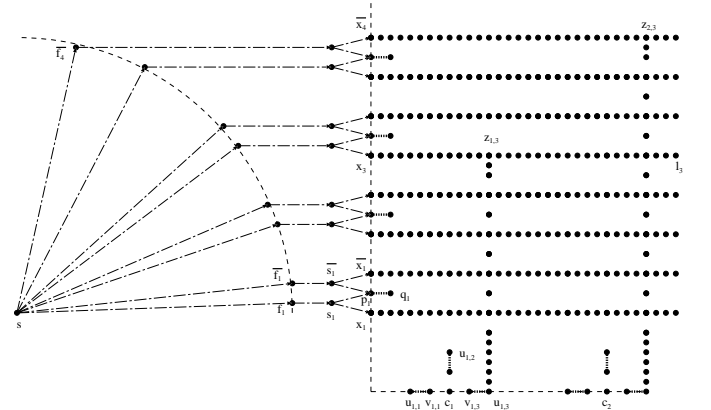


**Figure 4.**  $VFiller(a, b)$ . All nodes in the vertical line from  $a$  to  $b$  (except  $a$ ,  $b$ , and  $u$ ) belong to the  $VFiller$ . No nodes in the  $VFiller$  is within a unit distance from  $u$ . All nodes in the  $VFiller$  have a range of 1 except  $c$ , whose range is 4 (shown through the shaded circle).

**Source gadget:** The source gadget consists of the source node,  $s$  whose characteristic vector  $cv(s) = (-(range(s) +$

$8), x_1.y, 32(n + 3))$ . In addition, for every variable  $x_i \in X$ , we create two nodes  $f_i$  and  $\bar{f}_i$ . The nodes  $f_i$  and  $\bar{f}_i$  are placed on the arc of the top right quadrant of the circle, whose center is  $s$  and whose radius is  $range(s)$ . Also,  $f_i$  ( $\bar{f}_i$ ) is on the same horizontal line as  $s_i$  ( $\bar{s}_i$ ). Note that this uniquely defines the co-ordinates of  $f_i$  ( $\bar{f}_i$ ). We assign  $range(f_i) = d(f_i, s_i)$  and  $range(\bar{f}_i) = d(\bar{f}_i, \bar{s}_i)$ . This construction is illustrated in Figure 5. Specifically, see the part of figure which is to the left of the  $y$ -axis.

We now add more nodes to “connect” a clause to its literals and then we will specify  $range(u_{j_k}), k = 1, 2, 3$ . Consider a clause  $c_j \in C$ . Without loss of generality, let  $c_j = (x_{j_1} \vee x_{j_2} \vee x_{j_3})$ . For each  $k \in \{1, 2, 3\}$ , we do the following. Consider the node  $z_{j_k} \in Filler(x_{j_k}, l_{j_k}, 1)$  with co-ordinates  $(u_{j_k}.x, x_{j_k}.y)$ .  $z_{j_k}$  is the node which is vertically above  $u_{j_k}$  and belongs to  $Filler(x_{j_k}, l_{j_k}, 1)$ . We “connect”  $z_{j_k}$  with  $u_{j_k}$  using  $VFiller(z_{j_k}, u_{j_k})$ . Let  $a = D(x_{j_k}, z_{j_k}) = z_{j_k}.x - x_{j_k}.x$ .  $a$  is the length of the shortest path from  $x_{j_k}$  to  $z_{j_k}$ . Let  $b = D(z_{j_k}, u_{j_k}) = 1 + |VFiller(z_{j_k}, u_{j_k})|$ .  $b$  is the shortest path distance from  $z_{j_k}$  to  $u_{j_k}$ . Let  $p = L - 4 - a - b$ . Let  $\delta = d(u_{j_k}, v_{j_k})/p$ . We create  $Filler(u_{j_k}, v_{j_k}, \delta)$  and assign  $range(u_{j_k}) = \delta$ . Note that this choice of  $\delta$  makes the shortest path distance from any  $x_{j_k}$  to  $c_j$  equal to  $L - 3$ . This construction is illustrated in Figures 3 and 5. Recall that we set  $L = 32(m + n + 3)$ . This completes the reduction.



**Figure 5.** Reduction of the 3-SAT instance  $(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_4)$  to an *ad hoc* network minimum latency broadcast instance. Note how all the  $f_i$  nodes are along the arc. Node  $f_i$  can just reach  $s_i$ . Also note how the literals are connected to the clause nodes. For sake of clarity, we depict only the  $x_3 \rightsquigarrow c_1$ , and  $\bar{x}_4 \rightsquigarrow c_2$  connections and omit other literal-clause connections. Also, we omit many nodes in VFiller gadgets (for e.g., from  $z_{1,3}$  to  $u_{1,3}$ ).

## B. Properties of the network $\rho(I)$

**Property P0.** Consider nodes  $p_i$  and  $q_i$  created in the gadget for variable  $x_i$ .  $D(p_i, q_i) = L - 3$  and all paths from  $s$  to  $q_i$  are through  $p_i$ .

*Proof:* Recall that  $S = Filler(p_i, q_i, d(p_i, q_i)/(L - 3))$  is the set of nodes created in the gadget for the variable  $x_i$ .  $D(p_i, q_i) = |S| + 1 = L - 3$ . If  $u \notin S \cup \{p_i\}$ , then there is no arc from  $u$  to any node in  $S \cup \{q_i\}$ . Hence, all paths from  $s$  to  $q_i$  are through  $p_i$ . ■

**Property P1.** For any  $k \in \{1, 2, 3\}$ , consider the nodes  $u_{j_k}$  and  $v_{j_k}$  in the gadget for clause  $c_j$ . All paths from  $s$  to  $v_{j_k}$

are through  $u_{j_k}$ .

*Proof:* Recall that  $S = Filler(u_{j_k}, v_{j_k}, \delta)$  is the set of nodes created in gadget for clause  $c_j$ . Let  $w \notin \{u_{j_k}, v_{j_k}\} \cup S$ . Then, no node in  $S \cup \{v_{j_k}\}$  is in the transmission range of  $w$ , i.e., there is no edge from  $w$  to any node in  $S \cup \{v_{j_k}\}$ . Hence, all paths from  $s$  to  $v_{j_k}$  are through  $u_{j_k}$ . ■

**Property P2.** For  $k \in \{1, 2, 3\}$ , consider the nodes  $u_{j_k}$  and  $z_{j_k}$  in the gadget for clause  $c_j$ . All paths from  $s$  to  $u_{j_k}$  are through  $z_{j_k}$ .

*Proof:* Recall that  $S = VFiller(z_{j_k}, u_{j_k})$  is the set of nodes created in the gadget for clause  $c_j$ . Let  $w \notin \{z_{j_k}, u_{j_k}\} \cup S$ . Then,  $w$  does not have an edge to any node in  $S$ . Combining this with Property P1 proves the claim. ■

**Property P3:** Consider a node  $c_j$  corresponding to the clause  $(x_{j_1} \vee x_{j_2} \vee x_{j_3})$ . All paths from  $s$  to  $c_j$  contain some node in  $\{z_{j_1}, z_{j_2}, z_{j_3}\}$ .

*Proof:* Follows from Properties P1 and P2. ■

**Property P4.** Consider any node  $t \in Filler(x_i, l_i, 1)$ . Consider the path  $P$  from  $s$  to  $t$  which is the concatenation of paths  $(s, f_i, s_i)$  and the horizontal straight line path  $x_i \rightsquigarrow t$ . Consider any other path  $P'$  from  $s$  to  $t$  which does not go through  $x_i$ . Let  $|P|$  and  $|P'|$  denote the length of the paths.  $|P| \geq |P'| + 12$ . In particular, this property holds for  $t = z_{j_k}, k \in \{1, 2, 3\}$ , where  $z_{j_k}$  is a node considered in the gadget for clause  $c_j$ .

*Proof:* We first give the proof idea through Figures 4 and 5. Consider any fixed node  $t$  in a  $Filler(x_i, l_i, 1)$  in Figure 5. There is no node below  $t$  (i.e., whose  $y$ -coordinate is less than that of  $t$ ) whose range includes  $t$ . Hence, all paths from  $s$  to  $t$  are either through  $x_i$  or through a node in  $X$  which lies above  $x_i$  (and  $t$ ). For instance, let  $t = l_1$  in Figure 4 and let  $P'$  be  $s \rightsquigarrow \bar{x}_i \rightsquigarrow a \rightsquigarrow c \rightsquigarrow l_1$  (i.e.,  $P_1$  passes through  $\bar{x}_i$ ,  $a$ , and  $c$  to  $l_1$ ). Notice that the horizontal distance of  $\ell$  units from  $\bar{x}_1$  to  $l_1 = t.x - \bar{x}_1.x = t.x - x_1.x$ . Each hop can cover a horizontal distance of one unit, with the exception of the hop from  $c$ . This hop can reach the node which is three units to the right of  $c$ . However, it takes 14 vertical hops to get to  $c$  from  $a$ . Hence, the total number of hops from  $\bar{x}_1$  to  $l_1$  is  $\ell - 3 + 15 = \ell + 12$ , while the number of hops from  $x_1$  to  $l_1$  is exactly  $\ell$ .

In general,  $P'$  passes through some node  $r \in X - \{x_i\}$ . We have  $D(s, r) = D(s, x_i) = 3$ .  $P'$  should cover both the horizontal distance and the vertical distance between  $r$  to  $t$ . Horizontal distance between  $r$  and  $t = t.x - r.x = t.x - x_i.x$ . Thus the horizontal distance covered is the same in both  $P$  and  $P'$ . However, it takes at least an additional 12 hops to cover the vertical distance in  $P'$  between nodes  $r$  and  $t$  (by construction). ■

**Property P5.** Consider the clause node  $c_j$  corresponding to the clause  $(x_{j_1} \vee x_{j_2} \vee x_{j_3})$ . Let  $k \in \{1, 2, 3\}$ . Then  $D(x_{j_k}, c_j) = L - 3$ .

*Proof:* From Properties P1, P2, and P3, we have  $D(x_{j_k}, c_j) = D(x_{j_k}, z_{j_k}) + D(z_{j_k}, u_{j_k}) + D(u_{j_k}, v_{j_k}) + 1 = L - 3$ . ■

**Property P6.** Consider a clause node  $c_j$  corresponding to the clause  $(x_{j_1} \vee x_{j_2} \vee x_{j_3})$ . Consider a shortest path  $P$  from  $s$  to  $c_j$ .  $|P| = L$ , and  $P$  contains a node in  $\{x_{j_1}, x_{j_2}, x_{j_3}\}$ . Consider any path  $P'$  from  $s$  to  $c_j$  which does not contain any node in the  $\{x_{j_1}, x_{j_2}, x_{j_3}\}$ .  $|P'| \geq L + 12$ .

*Proof:* Follows from Properties P3, P4, and P5. ■

**Lemma 4.1:** If  $\rho(I)$  has a schedule of latency  $L$  then  $I$  is satisfiable.

*Proof:* Consider the set of nodes  $T \subset X$ , such that  $u \in T$  iff  $u$  receives the message collision-free at time 3. Assign all the literals corresponding to the nodes in  $T$  to be  $TRUE$  and those in  $X \setminus T$  to be  $FALSE$ . We will show that this assignment is both a valid and a satisfying assignment for  $I$ .

Consider any node  $q_i$ . Since  $q_i$  receives the message at time  $L$ , Property P0 implies that  $p_i$  receives the message at time 3. Hence, exactly one node in  $\{s_i, \bar{s}_i\}$  transmits the message at time 3. Thus exactly one literal in  $\{x_i, \bar{x}_i\}$  is assigned  $TRUE$ . Hence, the assignment is valid.

Consider a clause  $c_j$  corresponding to the clause  $(x_{j_1} \vee x_{j_2} \vee x_{j_3})$ . Since  $c_j$  gets the message by time  $L$ , Property P6 implies that at least one literal in  $\{x_{j_1}, x_{j_2}, x_{j_3}\}$  must receive the message at time 3 and hence is assigned  $TRUE$ . Hence, the assignment is a satisfying one. ■

**Lemma 4.2:** If  $I$  has a satisfying assignment then  $\rho(I)$  has a schedule of makespan  $L$ .

*Proof:* We now describe a broadcast schedule for the nodes in the network. Consider the set of literals  $T \subset X$  such that  $l \in T$  iff  $l$  is assigned  $TRUE$  in the satisfying assignment. For all  $i \in \{1, 2, \dots, n\}$ ,  $s_i$  transmits at time 3 and  $\bar{s}_i$  transmits at time 4 iff  $x_i \in T$ ; otherwise  $s_i$  transmits at time 4 and  $\bar{s}_i$  transmits at time 3. For all  $j \in \{1, 2, \dots, m\}$ , if more than one node in  $\{v_{j_1}, v_{j_2}, v_{j_3}\}$  receive the message by time  $L - 1$ , then one of them is arbitrarily chosen to transmit the message to  $c_j$  and the others do not transmit. Any other node  $u$ , which receives the message at time  $t$ , transmits the message at time  $t + 1$ .

We first show that this schedule is collision-free. Properties P0, P1 and P2 imply that there is no collision at any node which belongs to a set of the form  $VFiller(z_{j_k}, u_{j_k}, 1)$ ,  $Filler(p_i, q_i, \delta)$  or  $Filler(u_{j_k}, v_{j_k}, \delta)$ . For the same reason, if a node is of the form  $u_{j_k}$  or  $v_{j_k}$ , it does not experience collision. If a node is of the form  $f_i, s_i, x_i, q_i$ , or  $l_i$ , it does not experience collision (by construction, these nodes have a unique neighbor to receive the message from.) For any  $i$ ,  $p_i$  does not experience collision since exactly one node in  $\{s_i, \bar{s}_i\}$  transmits at time 3. No clause node  $c_j$  experiences collision since, exactly one of the three nodes in  $\{v_{j_1}, v_{j_2}, v_{j_3}\}$  is arbitrarily chosen to transmit. Finally, let  $r \in Filler(x_i, l_i, 1)$  for some  $i$ . Consider the unique shortest path  $P$  from  $s$  to  $r$  through  $x_i$ . The message reaches  $r$  along this path by time  $D(s, r) + 1$ . By Property P4, any other path from  $s$  to  $r$  has length  $\geq D(s, r) + 12$  and hence cannot cause collision at  $u$ .

We now show that the schedule has a latency of  $L$ . Consider any node  $q_i$ .  $p_i$  receives the message at time 3. Hence  $q_i$  receives the message at time  $D(p_i, q_i) + 3 = L$ . Consider any clause node  $c_j$ . It has at least one literal  $x_{j_k}$  which is assigned  $TRUE$ . Consider the shortest path of length  $L$  from  $s$  to  $c_j$  through  $x_{j_k}$ . Every node  $u$  in this path transmits the message at time  $D(s, u)$ . Hence,  $c_j$  receives the message at time  $D(s, c_j) = L$ . Consider any other node  $u$  which is not of the form  $q_i$  or  $c_j$ . Consider the shortest path from  $s$  to  $u$ . Every node  $v$  in this path transmits the message by time  $D(s, v) + 1$ . Since  $D(s, u) \leq L - 1$ ,  $u$  receives the message

by time  $D(s, u) + 1 \leq L$ . ■

Lemmas 4.1 and 4.2 together imply the following theorem.

*Theorem 4.3:* The minimum latency broadcast problem for ad hoc networks is NP-Complete.

## V. BROADCAST ALGORITHMS

### A. Scheduling Broadcasts for a Single Message

The algorithm takes as input a directed graph  $G = (V, E)$  and a source node  $s$ . Let  $N_i(u)$  and  $N_o(u)$  denote the set of in-neighbors and out-neighbors of a node  $u \in V$ . 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 broadcast tree  $T_b$ , is constructed as follows (pseudocode BROADCASTTREE given below). The set of nodes  $V$ , is partitioned into *primary nodes*,  $P$ , and *secondary nodes*  $S$ . 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  $P$ . If  $G$  is undirected,  $P$  is any maximal independent set in  $G$ . However, if  $G$  is directed, a maximal independent set need not be a dominating set. Hence,  $P$  is constructed as follows. Initially,  $P$  contains only  $s$ . The nodes in  $V$  are processed in the increasing order of their levels in the Breadth First Search (BFS) tree that is rooted at  $s$ . A node  $w \in V$  is added to  $P$  iff  $P$  constructed so far does not dominate  $w$ . For each node  $u \in V$ , its parent,  $parent(u)$ , in  $T_b$ , is determined as follows. Let  $L_i, i = 0, 1, 2, \dots, l$ , be the set of nodes at level  $i$  in the BFS tree. 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. If  $u \in P_i$  then  $parent(u)$  is any one of its in-neighbors in  $L_{i-1}$ . If  $u \in S_i$  then  $parent(u)$  is any one of its in-neighbors in  $P_{i-1} \cup P_i$ . All nodes whose parent is  $u$  constitute the set  $C(u)$  (children of  $u$ ). The running time of this algorithm is dominated by the time required to compute the BFS tree. Hence it runs in linear time.

BROADCASTTREE( $G = (V, E), s$ )

```

1   $P \leftarrow \{s\}$  //  $P$  is the set of primary nodes.
2   $T_{BFS} \leftarrow$  BFS tree in  $G$  with root  $s$ 
3   $l \leftarrow$  maximum number of levels in  $T_{BFS}$ 
   //  $s$  belongs to level 0
4  for  $i \leftarrow 1$  to  $l - 1$  do
5       $L_i \leftarrow$  set of all nodes at level  $i$  in  $T_{BFS}$ 
6      for each  $w \in L_i$  do
7          if  $(P \cap N_i(w) = \emptyset)$  then
8               $P \leftarrow P \cup \{w\}$ 
9               $parent(w) \leftarrow L_{i-1} \cap N_i(w)$ 
10         else //  $w$  is a secondary node
11              $parent(w) \leftarrow$  any node in  $P$  dominating  $w$ 
12     for each  $u \in V$  do
13          $C(u) \leftarrow \{w | parent(w) = u\}$ 
14      $V_b \leftarrow V$ 
15      $E_b \leftarrow \{(u, w) | u = parent(w)\}$ 
16     return  $T_b = (V_b, E_b)$ 

```

In SCHEDULEBROADCASTS (pseudocode given below), the transmissions are scheduled following a *greedy* strategy. Note that the nodes that transmit the message are the internal (non-leaf) nodes in  $T_b$ . Any transmitting node,  $u$ , transmits at the minimum time  $t$  that satisfies the following three constraints – (i)  $u$  has received the message collision-free before time  $t$ , (ii) no node in  $C(u)$  is hearing any transmissions at time  $t$ , (iii) no node in  $N_o(u) \setminus C(u)$  is receiving the message collision-free at time  $t$ , where  $N_o(u)$  is the set of out-neighbors of  $u$ . Note that line 6 in our algorithm simply checks for (ii), line 7 checks for (iii), and line 9 chooses the minimum time such that all three constraints above are satisfied. This algorithm sets the transmit time for each node in the graph at most once, at which time it examines all its out neighbors. Hence, this algorithm also runs in linear time.

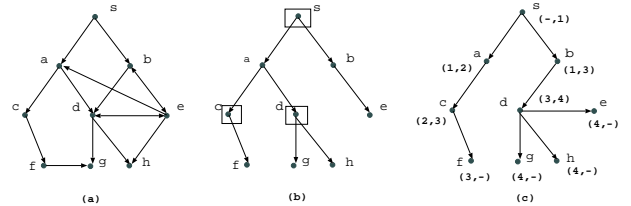
SCHEDULEBROADCASTS( $T_b, s$ )

```

1  for each node  $u \in T_b$  do
2       $trTime(u) \leftarrow 0$ 
3   $Transmitters \leftarrow \{s\}$ 
4  while  $(Transmitters \neq \emptyset)$  do
5       $u \leftarrow$  any node in  $Transmitters$ 
6       $I_1(u) \leftarrow \{t | \exists w \in C(u)$ 
   // that hears a msg at time  $t\}$ 
7       $I_2(u) \leftarrow \{t | \exists w \in N_o(u)$ 
   // that receives
   // a msg. coll-free at time  $t\}$ 
8       $I(u) \leftarrow I_1(u) \cup I_2(u)$ 
9       $trTime(u) \leftarrow \min\{t | t > rcvTime(u)$ 
   // and  $t \notin I(u)\}$ 
10     for each  $w \in C(u)$  do
11          $rcvTime(w) \leftarrow trTime(u)$ 
12          $Transmitters \leftarrow Transmitters \setminus \{u\}$ 
13          $Transmitters \leftarrow Transmitters \cup$ 
   //  $\{w | w \in C(u) \text{ and } C(w) \neq \emptyset\}$ 
14     return  $trTime$ 

```

The figure below (Figure 6) illustrates our algorithm.



**Figure 6.** An illustration of our algorithm. (a) shows the example network, (b) shows the BFS tree,  $T_{BFS}$  along with the primary nodes (highlighted), and (c) shows  $T_b$ . Arcs are shown from parent to its children. Besides each node are shown its  $rcvTime$  and  $trTime$ .

### B. Analysis

*Lemma 5.1:*  $T_b$  is a connected tree rooted at  $s$ .

*Proof:* We show that every node in  $V$  has exactly one parent in  $T_b$ , with the exception of  $s$  which has none; this implies the lemma immediately. Since  $s \in L_0$ , it is never considered by the **for** loop in line 4 and hence does not have a parent. Consider any node  $v \neq s$ . Let  $v \in L_i$ . There are two cases. In the first case, assume  $v$  is a primary node. Line 9 ensures that it has a unique parent in the set  $L_{i-1}$ . In the

second case, assume  $v$  is a secondary node. Clearly, for this to happen some primary node in  $L_{i-1}$  or  $L_i$  must dominate  $v$ ; otherwise,  $v$  would have been designated as a primary node (see the **for** loop in line 6). Thus,  $P_{i-1} \cup P_i$  is non-empty and hence  $v$  has a unique parent in the tree (line 11). ■

*Lemma 5.2:* For any two primary nodes  $u$  and  $w$  in  $P$ ,  $d(u, w) > r_{min}$ .

*Proof:* Without loss of generality, assume that  $u$  was chosen in  $P$  before  $w$ . If  $d(u, w) \leq r_{min}$  then  $w$  would be dominated by  $u$  and would not have been chosen in  $P$  by our algorithm. Notice that while the proof of this lemma is easy, this property comes about due to the specific way in which we choose primary nodes in our algorithm, and in general does not hold if the broadcast trees are not chosen carefully. This property will be useful in bounding both the latency and the number of transmissions later. ■

*Lemma 5.3:* If  $(u, w) \in E_b$  then both  $u$  and  $w$  cannot be secondary nodes, i.e.,  $|\{u, w\} \cap P| \geq 1$ .

*Proof:* Without loss of generality, let  $u = \text{parent}(w)$ . If  $w \in P$  then the claim is true. If  $w$  is a secondary node then  $w$  chooses a primary node as its parent in line 11 of BROADCASTTREE. Hence,  $u$  must be a primary node. ■

*Lemma 5.4:* Every node transmits at most once.

*Proof:* Consider any node  $v \in V$ . If  $v$  is a leaf in  $T_b$  then  $C(v) = \emptyset$  and  $v$  is never included in the list *Transmitters* (line 13 in SCHEDULEBROADCASTS). Hence,  $v$  never transmits. If  $v$  is an internal node in  $T_b$  then it transmits exactly once at  $\text{trTime}(v)$ . Node  $v$  is then removed from *Transmitters* (line 12 in SCHEDULEBROADCASTS). ■

*Lemma 5.5:* Consider a disk  $D$  of radius  $r \geq r_{min}$ . If  $\text{Primaryes}(r)$  represent the set of primary nodes within  $D$  then  $|\text{Primaryes}(r)| \leq k_1 r^2 / r_{min}^2$ , where  $k_1$  is a constant.

*Proof:* Let  $c$  be the center of the disk  $D$ . For each primary node  $u \in \text{Primaryes}(r)$ , consider a disk of radius  $r_{min}/2$  centered at  $u$ . Let  $H$  denote the set of all such disks. Thus,  $|\text{Primaryes}(r)| = |H|$ . By Lemma 5.2, no two primary nodes are within distance  $r_{min}$  from each other. Thus,  $H$  is a set of non-intersecting disks. Each primary node in  $D$  is at most a distance of  $r$  from  $c$ . Any point on a disk in  $H$  is at a distance of at most  $r + (r_{min}/2)$  from  $c$ . The size of  $H$  is upper-bounded by the number of non-intersecting disks of radius  $r_{min}/2$  whose areas are completely contained within a disk of radius  $r + (r_{min}/2)$ . Hence we have  $|H| \leq \pi(r + (r_{min}/2))^2 / (\pi(r_{min}/2)^2) = ((2r/r_{min}) + 1)^2 \leq (3r/r_{min})^2$ . ■

*Lemma 5.6:* Consider an internal node  $u$  in  $T_b$  that has received the message collision-free. Recall that  $I(u)$  (line 8 in SCHEDULEBROADCASTS) is the set of times that  $u$  must avoid while choosing  $\text{trTime}(u)$ . Then  $|I(u)| \leq 2|\text{Primaryes}(3r_{max})| \leq k_2 r_{max}^2 / r_{min}^2$ , where  $k_2$  is a constant.

*Proof:* Let us represent the transmission range of any node  $u$  by a disk of radius  $\text{range}(u) \leq r_{max}$ . Let  $w$  be a transmitter such that  $u$  and  $w$  share a common out-neighbor and  $\text{trTime}(w) \in I(u)$ . Thus,  $d(u, w) \leq 2r_{max}$ . Let  $B(u)$  be the set of all such nodes  $w$ . Since each node transmits at most once,  $|I(u)| \leq |B(u)|$ . We will now upper-bound  $|B(u)|$ . Nodes in  $B(u)$  are either primary nodes,  $B^p(u)$ , or

secondary nodes,  $B^s(u)$ . Let us first bound  $|B^p(u)|$ . Note that  $|B^p(u)| \leq |\text{Primaryes}(2r_{max})|$ , where we can obtain  $|\text{Primaryes}(2r_{max})|$  from Lemma 5.5. We will now bound  $|B^s(u)|$ . Let  $v \in B^s(u)$ . By Lemma 5.3, every child of  $v$  in  $T_b$  is a primary node. These primary nodes are at a distance of at most  $3r_{max}$  from  $u$ . Thus,  $|B^s(u)| \leq |\text{Primaryes}(3r_{max})|$ . Using Lemma 5.5, we get  $|B^p(u)| + |B^s(u)| \leq 2|\text{Primaryes}(3r_{max})| \leq k_2 r_{max}^2 / r_{min}^2$ , for some constant  $k_2$ . ■

1) *Latency Bound:* Recall that  $l$  is the depth of  $T_{BFS}$ . Let  $OPT_{lat}$  be the optimal broadcast latency. Then  $l \leq OPT_{lat}$ .

*Theorem 5.7:* Algorithm BROADCASTMESSAGE gives an approximation guarantee of  $O(r_{max}^2 / r_{min}^2)$ .

*Proof:* Recall that for any node  $u$ ,  $\text{rcvTime}(u)$  is the time when  $u$  receives the message collision-free from its parent in  $T_b$ . By induction on the number of levels in the BFS tree, we show that for any  $i = 0, 1, \dots, l$ , and some const.  $k$ ,

$$\forall v \in L_i, \text{rcvTime}(v) \leq t_i = k(r_{max}^2 / r_{min}^2) i \quad (1)$$

Then substituting  $i = l$  and using  $l \leq OPT_{lat}$  proves the theorem. We now prove (1).

Expression (1) is true for  $i = 0$ , since  $L_i = \{s\}$  and by definition,  $s$  receives the message at time 0. We now assume that (1) is true for  $i = 0, 1, \dots, q$  and show that (1) is true for  $i = q+1$ . Let  $z \in L_{q+1}$ . Let  $y = \text{parent}(z)$  and  $x = \text{parent}(y)$ . A node  $u$  may transmit the message only after it receives the message collision-free. It also avoids all times in  $I(u)$  to schedule the transmission. After satisfying these constraints  $u$  chooses  $\text{trTime}(u)$  greedily. Thus,  $\text{rcvTime}(z) \leq \text{rcvTime}(y) + 1 + |I(y)| \leq \text{rcvTime}(x) + 2 + |I(y)| + |I(x)|$ . Note that either  $x \in L_{q-1}$  or  $x \in L_q$ . Using Lemma 5.6, induction hypothesis, and choosing  $k \geq 4k_2$ , we get  $\text{rcvTime}(z) \leq t_q + 2 + 2k_2(r_{max}^2 / r_{min}^2) \leq k(r_{max}^2 / r_{min}^2)q + 2 + 2k_2(r_{max}^2 / r_{min}^2) \leq k(r_{max}^2 / r_{min}^2)(q + 1)$ . ■

2) *Bound on the number of transmissions:* Let  $OPT_{ret}$  be the set of nodes that transmit in an optimal algorithm.

*Lemma 5.8:* The total number of transmissions in our algorithm is at most  $2|P|$ .

*Proof:* In our algorithm, each node transmits at most once (Lemma 5.4). Hence, the total number of transmissions is at most  $|P| + |S_{ret}|$ , where  $P$  is the set of primary nodes and  $S_{ret}$  is the set of transmitting secondary nodes. By Lemma 5.3, in  $T_b$ , every child of a secondary node is a primary node with a unique parent. Hence,  $|S_{ret}| \leq |P|$ . ■

*Lemma 5.9:* Recall that  $\text{Primaryes}(r_{max})$  is the set of primaries in a disk of radius  $r_{max}$ . Then we have  $|OPT_{ret}| \geq |P| / |\text{Primaryes}(r_{max})|$ .

*Proof:* Any node has a maximum transmission range of  $r_{max}$ . The maximum number of primary nodes in a disk of radius  $r_{max}$  is  $|\text{Primaryes}(r_{max})|$ . Since each node in  $OPT_{ret}$  can reach at most  $|\text{Primaryes}(r_{max})|$  primary nodes, we get the required bound. ■

*Theorem 5.10:* The number of transmissions in our algorithm is  $O(r_{max}^2 / r_{min}^2)$  times the number of transmissions in an optimal algorithm.

*Proof:* Follows from Lemmas 5.8, 5.9, and 5.5. ■

*Corollary 5.11:* If the minimum and maximum node transmission ranges are bounded then the latency and the number

of transmissions in our algorithm are at most  $O(1)$  times their respective optimal values.

### C. Algorithms for Scheduling Multiple Messages

We present three algorithms for scheduling broadcasts of multiple messages. The first is a “one-to-all” broadcast algorithm which schedules the broadcast of multiple messages from a single source. The second and third are “all-to-all” broadcast algorithms which schedule the broadcast of multiple messages from many sources. For the remainder of the section, we assume that the network has a uniform transmission range, i.e.,  $r_{min} = r_{max}$ . Recall that  $M = \{1, \dots, m\}$  is the set of messages. We now present our algorithm for scheduling the broadcast of messages in  $M$  from a single source  $s$ .

1) *Single Source Multiple Messages (SSMM)*: The algorithm (pseudocode SCHEDULEBROADCASTS\_SSMM given below) consists of three stages. The first stage computes the broadcast tree from  $s$  using the algorithm BROADCASTTREE in Section V. The second stage computes a schedule for a single message using a modified version of the scheduling algorithm SCHEDULEBROADCASTS. The third stage efficiently computes a schedule for every message in  $M$ . We now present the details of the last two stages.

Stage 2 is identical to the algorithm SCHEDULEBROADCASTS except Line 9 which is replaced by lines 9.1, 9.2 and 9.3 as shown in the pseudo-code below. Recall that  $L_i$  is the set of nodes at level  $i$  in the BFS tree rooted at  $s$ . Let  $t_i = ki$ . Line 9.2 computes the  $trTime$  of a node  $u \in L_i$ . We will ensure that any node in  $L_i$  transmits the message between times  $t_i + 1$  and  $t_{i+1}$  (Lemma 5.12). Line 9.3 specifies  $delay(u)$ , which is the amount of time by which  $u$  delays the transmission of a message after receiving it.

In Stage 3,  $s$  transmits messages 1 through  $m$  sequentially with a uniform gap of  $3k$  time units between two successive messages, where  $k$  is the constant appearing in expression (1). Specifically, message  $j$  is transmitted by  $s$  at time  $3(j-1)k+1$ . Any transmitting node  $u$  which receives message  $j$ , delays the transmission of  $j$  by  $delay(u)$  time units computed in line 9.3.

SCHEDULEBROADCASTS\_SSMM( $G = (V, E)$ ,  $M$ ,  $s$ )

#### Stage 1.

$T_b \leftarrow \text{BROADCASTTREE}(G, s)$

#### Stage 2.

// Ident. to SCHEDULEBROADCASTS( $\cdot, \cdot$ ) except  
 // line 9 which is replaced by the next three lines  
 9.1 Let  $u \in L_i$   
 9.2  $trTime(u) \leftarrow \min\{t | t > t_i \text{ and } t > rcvTime(u) \text{ and } t \notin I(u)\}$   
 9.3  $delay(u) = trTime(u) - rcvTime(u)$

#### Stage 3.

**for**  $j \leftarrow 1$  to  $m$  **do**  
   **for** each node  $u \in T_b$  **do**  
     **if**  $u = s$  **then**  
       transmit msg.  $j$  at time  $3(j-1)k+1$   
     **else**  
       after receiving message  $j$ , delay transmission of  $j$  by  $delay(u)$  time units  
**return** all transmit times

Why should  $s$  introduce a gap of  $3k$  time units between two successive message transmissions? To answer this question, we make the following observations. Let  $L_i$  and  $L_{i'}$  be two BFS levels such that  $i' > i$ . Our first observation is that, in a network with only bi-directional edges,  $L_i$  and  $L_{i'}$  have an edge between them only if  $i' \leq i + 1$ . Specifically, transmissions from  $L_i$  to  $L_{i+1}$  and transmissions from  $L_{i+3}$  to  $L_{i+4}$  can proceed simultaneously without interfering with each other. Thus, it suffices to “separate” two different messages by three levels in order to allow simultaneous collision-free transmissions of these messages. We also show in Lemma 5.12 that all nodes in  $L_i$  transmit the first message during the  $k$  time units between time  $t_i + 1$  and  $t_{i+1}$ . Clearly,  $3k$  time units is an appropriate choice of the gap between successive messages.

2) *Multiple Source Multiple Messages*: We present two algorithms for scheduling broadcasts of multiple messages with multiple sources. Recall that for every message  $j \in M$ ,  $s_j$  is the source of message  $j$ . In the first algorithm, called Intermediate Source Broadcast (ISB), messages get unicast from their sources  $s_j$  to an arbitrarily chosen intermediate source  $s$ . These unicasts themselves are greedily scheduled to ensure collision-free transmissions (see section V-C.3). After  $s$  receives all the messages, it broadcasts them using the algorithm SCHEDULEBROADCASTS\_SSMM. In the second algorithm called Multi-Source Broadcast (MSB), a broadcast tree  $T_j$  is constructed from each source  $s_j$ . Message  $j$  gets broadcast on tree  $T_j$  using the greedy scheduling algorithm SCHEDULEBROADCASTS. We now briefly explain the details of the algorithms below.

3) *Intermediate Source Broadcast (ISB)*: This algorithm consists of two stages. Consider the shortest path  $p_j$  from  $s_j$  to  $s$ . In the first stage, every message  $j$  gets unicast along  $p_j$  to  $s$ . We ensure that these unicasts are collision-free by a greedy scheduling strategy that gives priority to message  $j_1$  over  $j_2$  iff  $j_1 < j_2$ . In the second stage,  $s$  uses SCHEDULEBROADCASTS\_SSMM to broadcast all the messages.

4) *Multi-Source Broadcast (MSB)*: In the multi-source broadcast algorithm, each source  $s_j$  of the message computes a broadcast tree rooted at itself using the algorithm BROADCASTTREE. Message  $j$  is broadcast along tree rooted at  $s_j$ . Transmissions are scheduled using the algorithm SCHEDULEBROADCASTS. Message  $j_1$  gets priority over  $j_2$  iff  $j_1 < j_2$ . This priority is enforced by scheduling  $j_1$  before  $j_2$ .

The two algorithms for multiple messages present interesting trade offs. ISB requires only one broadcast tree to be computed and maintained at  $s$ , apart from the unicasts from each  $s_j$  to  $s$ . MSB, on the other hand, requires computation and maintenance of potentially  $m$  trees from each  $s_j$ . However, intuitively, MSB should have a lower broadcast latency, since the messages are not routed through an intermediate source. Experimental studies (see section VII-B) confirm this intuition.

### D. Analysis for Multiple Messages

1) *Latency bound for SSMM*: Let  $OPT_{lat}$  be the latency for broadcasting the messages in an optimal solution. Recall that  $k$  is a constant which appears in (1). We will prove that the latency is at most  $4kOPT_{lat}$ .

*Lemma 5.12:* Consider Stage 2 of the algorithm SCHEDULEBROADCASTS\_SSMM in which we compute the schedule for a single message. Let  $u \in L_i$ . Then we have

$$t_{i-1} < rcvTime(u) \leq t_{i+1} \quad (2)$$

$$t_i < trTime(u) \leq t_{i+1} \quad (3)$$

*Proof:* The earliest that any node  $u \in L_i$  can receive a message is when some node in  $L_{i-1}$  transmits a message. Line 9.2 in Algorithm SCHEDULEBROADCASTS\_SSMM ensures that this happens only after time  $t_{i-1}$ . This proves the lower bound on  $rcvTime(u)$  in (2). Line 9.2 also ensures that  $u$  transmits only after time  $t_i$ . This proves the lower bound on  $trTime(u)$  in (3). We will now prove the upper bounds in (2) and (3) by induction on  $i$ . Expressions (2) and (3) are true when  $i = 0$ , since  $trTime(s) = 1$  and  $rcvTime(s) = 0$ . Assume that (2) and (3) hold for  $i = q$ . We will prove that (2) and (3) hold for  $i = q + 1$ . We will use the facts that for any node  $v$ ,  $|I(v)| \leq k_2$  (Lemma 5.6) and  $k \geq 4k_2$  (proof of Theorem 5.7). Let  $u \in L_{q+1}$ . We now consider the following two cases. Case 1:  $u$  is a primary node. Hence,  $parent(u) \in L_q$ . Hence, by induction hypothesis,  $rcvTime(u) = trTime(parent(u)) \leq t_{q+1}$ .  $trTime(u) \leq t_{q+1} + 1 + |I(u)| \leq t_{q+1} + 1 + k_2 < t_{q+1} + k = t_{q+2}$ . Case 2:  $u$  is a secondary node.  $parent(u) \in L_i \cup P_{i+1}$ . Hence,

$$\begin{aligned} rcvTime(u) &= trTime(parent(u)) \\ &\leq t_{q+1} + 1 + |I(parent(u))| \\ &\leq t_{q+2}. \end{aligned} \quad (4)$$

$trTime(u) \leq \max\{t_{q+1}, rcvTime(u)\} + 1 + |I(u)| \leq t_{q+1} + 1 + |I(parent(u))| + 1 + |I(u)|$  (using (4))  $\leq t_{q+1} + 2 + 2k_2 < t_{q+1} + 4k_2 \leq t_{q+1} + k = t_{q+2}$ . ■

Let  $rcvTime(u, j)$  and  $trTime(u, j)$  be the times at which  $u$  receives and transmits message  $j$ .  $\forall u \in L_q$ , the following lemma holds.

*Lemma 5.13:* If  $u \in L_i$  then

$$\begin{aligned} 3k(j-1) + t_{q-1} &< rcvTime(u, j) \leq 3k(j-1) + t_{q+1} \\ 3k(j-1) + t_q &< trTime(u, j) \leq 3k(j-1) + t_{q+1} \end{aligned}$$

*Proof:* Source  $s$  delays the transmissions of successive messages by  $3k$  time units. This along with Lemma 5.12 give the lower bounds on  $rcvTime(u, j)$  and  $trTime(u, j)$ . We will prove the upper bound on  $trTime(u, j)$  which will imply the upper bound on  $rcvTime(u, j)$ .  $trTime(u, j) = 3k(j-1) + trTime(u, 1) = 3k(j-1) + trTime(u)$ . This along with Lemma 5.12 proves the claim. ■

*Lemma 5.14:* SCHEDULEBROADCASTS\_SSMM produces a collision-free schedule.

*Proof:* Assume otherwise. Let  $u_1 \in L_{i_1}$  and  $u_2 \in L_{i_2}$  transmit messages  $j_1$  and  $j_2$  respectively which results in a collision at node  $w$ . Let  $trTime(u_1)$  be chosen before  $trTime(u_2)$  in line 9.2 in Algorithm SCHEDULEBROADCASTS\_SSMM. We consider the following two cases. Case 1:  $j_1 = j_2$ . In this case,  $trTime(u_1) \in I(u_2)$ . Line 9.2 in SCHEDULEBROADCASTS\_SSMM ensures that  $trTime(u_2) \neq trTime(u_1)$ , a contradiction. Case 2:  $j_1 \neq j_2$ . In this case, Lemma 5.13 implies that there exist constants  $d_1$  and  $d_2$ ,  $1 \leq d_1, d_2 \leq k$  such that,  $trTime(u_1, j_1) = 3k(j_1-1) + ki_1 + d_1$ ,  $trTime(u_2, j_2) = 3k(j_2-1) + ki_2 + d_2$ ,

which implies,  $3k(j_2-j_1) = k(i_1-i_2) + d_1 - d_2$ . Transmissions by  $u_1$  and  $u_2$  can collide at a node only if they are at most two BFS levels apart, i.e.,  $|i_1 - i_2| \leq 2$ . Since,  $|j_2 - j_1| \geq 1$  and  $|d_1 - d_2| < k$ , this is a contradiction. ■

*Theorem 5.15:* Algorithm SSMM produces a collision-free schedule of latency  $Z_{SSMM} \leq 4kOPT_{lat}$ .

*Proof:* Let  $v \in L_l$  be the last node to receive any message collision free. Thus,  $Z_{SSMM} = rcvTime(v, m)$ . From Lemma 5.13 and using the fact that  $m$  and  $l$  are both lower bounds on  $OPT_{lat}$ , we get  $Z_{SSMM} \leq 3k(m-1) + k(l+1) \leq 3km + kl \leq 4kOPT_{lat}$ . This along with Lemma 5.14 proves the claim. ■

2) *Latency bound for ISB:* Let  $Z_{uni}$  and  $Z_{br}$  be the latency of the first and second stages of ISB respectively. Since the second stage of ISB is the same as SSMM,  $Z_{br} = Z_{SSMM}$ . Recall that  $s_j$  is the source of message  $j$ . Let  $L_0, L_1, \dots, L_l$  be the levels in the BFS tree rooted at  $s$ . For all  $j$ , let  $s_j \in L_{q_j}$ .

*Lemma 5.16:* All messages reach  $s$  without collision by time  $Z_{uni} \leq 3(m-1) + l$

*Proof:* Let  $f(j, i)$  denote that time at which the message  $j$  reaches level  $i$  without collision in the first stage. If  $(i \geq q_j)$  or  $(i < 0)$ , then  $f(j, i)$  is assumed to be 0. We now prove by induction on  $j$  and  $i$  that:

$$\forall j \in \{1, 2, \dots, m\}, \forall i, f(j, i) \leq 3(j-1) + (l-i) \quad (5)$$

When  $j = 1$ , the message gets the highest priority. Message 1 starts at time 1 from  $s_1$  and reaches  $s$  at time  $q_1$ . Hence (5) holds. Assume (5) holds  $\forall j \leq j'$ . For  $j = j' + 1$ , assume (5) holds for  $i = i' + 1$ . We prove (5) for  $j = j' + 1$  and  $i = i'$ . By the induction hypothesis, we have  $f(j'+1, i'+1) \leq 3j' + (l-1-i')$ ,  $\forall j \leq j'$ ,  $f(j, i'-2) \leq 3(j-1) + (l-(i'-2)) \leq 3j' + (l-1-i')$ . Hence, message  $j' + 1$  can reach level  $i'$  in the greedy unicast schedule, without collision, by time  $f(j'+1, i') \leq 3j' + (l-i')$ . ■

*Theorem 5.17:* Algorithm ISB produces a collision-free schedule with latency  $T_{ISB}$  such that

$$3k(m-1) + k(l-1) < Z_{ISB} \leq 6kOPT_{lat} \quad (6)$$

*Proof:* We first note that since both the stages of ISB are collision-free, the resultant schedule is collision-free. Let  $levels(u)$  denote the depth of the BFS tree (number of levels  $-1$ ) rooted at node  $u$ . Let  $l' = \max\{levels(s_j) | j \in M\}$ . Thus,  $l' \leq OPT_{lat}$ . If  $w$  is the farthest node from  $u$  then for any node  $v$ , we have  $D(u, w) \leq D(u, v) + D(v, w)$ . Hence,  $levels(u) \leq 2levels(v)$ . In particular, this implies that  $l \leq 2l'$ . Thus,  $Z = Z_{uni} + Z_{br} \leq 3(m-1) + l + 3km + kl \leq (3k+3)m + (k+1)l \leq (3k+3)m + 2(k+1)l' \leq (5k+5)OPT_{lat} \leq 6kOPT_{lat}$ . The lower bound follows from the lower bound on  $Z_{br}$  which follows from the first expression of Lemma 5.13. ■

3) *Bound on the Number of transmissions:* We now prove the bound on the number of transmissions for our multiple message broadcast algorithms. The bounds below hold for networks with uni-directional edges also.

*Theorem 5.18:* The number of transmissions in SSMM and MSB is  $O(r_{max}^2/r_{min}^2)$  times the number of transmissions in the optimal algorithm.

*Proof:* Theorem 5.10 guarantees that the number of transmissions for a particular message in SSMM and MSB

is  $O(r_{max}^2/r_{min}^2)$  times the number of transmissions for that message in the optimal algorithm. Adding number of transmissions for all messages in SSMM, MSB and in the optimal algorithm proves this theorem. ■

*Theorem 5.19:* The number of transmissions in ISB is  $O(r_{max}^2/r_{min}^2)$  times the number of transmissions in the optimal algorithm.

*Proof:* Let  $R_{uni}$  and  $R_{br}$  denote the number of transmissions during the first stage and second stage of ISB. The total number of transmissions in ISB is  $R_{uni} + R_{br}$ . Let  $OPT_{ret}$  denote the number of transmissions required to broadcast all the messages in an optimal solution. During the first stage, all messages reach  $s$  along their shortest paths. Hence,  $R_{uni} \leq OPT_{ret}$ . Recall that the proof of Theorem 5.10 holds regardless of the source of the message in an optimal algorithm. Thus, adding the number of transmissions for all messages in ISB and the optimal algorithm, we have  $R_{br} = O(r_{max}^2/r_{min}^2)OPT_{ret}$ . This completes the proof. ■

*Corollary 5.20:* For networks with bounded maximum and minimum node transmission ranges, the number of transmissions in all three multiple message broadcast algorithms is  $O(1)$  times the number of transmissions in an optimal solution.

## VI. DISTRIBUTED IMPLEMENTATION

The algorithms presented in this paper have easy distributed implementations when all the edges in the network are bi-directional. Our distributed implementations are inspired by the technique presented in [22]. The basic building blocks for our distributed implementation are the Depth First Search (DFS) and the Breadth First Search (BFS) graph traversal algorithms for which distributed implementations are available.

In order to build the broadcast tree, the message source creates a token. The purpose of this token is to record the information about primary nodes and their parents. This token visits each node of the graph during its DFS traversal. When a node receives the token, it knows the current set of primaries and their parents. In particular, it can determine if it is within the range of any current primary. If not, it adds itself to the set of primaries, chooses its parent, updates this information in the token, and passes the token to the next node in the DFS traversal. Since each edge is traversed twice in a DFS traversal, once forward and once backward, a node also computes its set of children as the token visits it through the back edges.

In order to compute the schedule for a message, the source creates another token. The purpose of this token is to record the information about the receive and transmit times of the nodes in the graph. This token does a DFS traversal of the non-leaf nodes of the broadcast tree created above. When a node receives this token, it knows the transmit times of all the internal nodes visited so far. In particular, it knows the transmit times of the nodes which could interfere with its own transmission. The node now computes its minimum possible collision-free transmit time, updates this information in the token and sends it to the next node to be visited. In other words, the single message broadcast algorithm can be implemented using two DFS traversals.

We now discuss the distributed implementations of our multiple message broadcast algorithms. We note that essentially the same technique discussed above can be applied for the distributed implementation of the single source multiple message (SSMM) algorithm. The first DFS computes the broadcast tree from the source  $s$ . In the second DFS, every transmitting node  $u$  computes  $delay(u)$ , the time by which it should delay the transmission of any message after receiving it. After this step,  $s$  starts transmitting all the messages with a uniform gap of  $3k$  time units between two successive messages (see section V-C.1). In the intermediate source broadcast (ISB), the second stage is the same as SSMM. For the first stage, ISB computes the BFS tree rooted at  $s$ . At the end of this computation, every node knows its parent and its level in the BFS tree and the maximum number of levels  $l$  in the BFS tree. Let  $u \in L_i$  be a node in the shortest path from  $s_j$  to  $s$ .  $u$  transmits the unicast message  $j$  at time  $f(u, j) = 3(j-1) + (l-i)$ . For the multiple source broadcast (MSB), we perform a DFS from each source  $s_j$  to compute its broadcast tree and for each message  $j$  to compute its schedule. This is precisely the trade-off referred to earlier between ISB and MSB: ISB involves two DFS traversals and one BFS traversal which is independent of the number of messages  $m$ , whereas MSB involves potentially  $2m$  DFS traversals. However, MSB has a much better broadcast latency compared to ISB (see Section VII-B).

## VII. EXPERIMENTAL EVALUATION

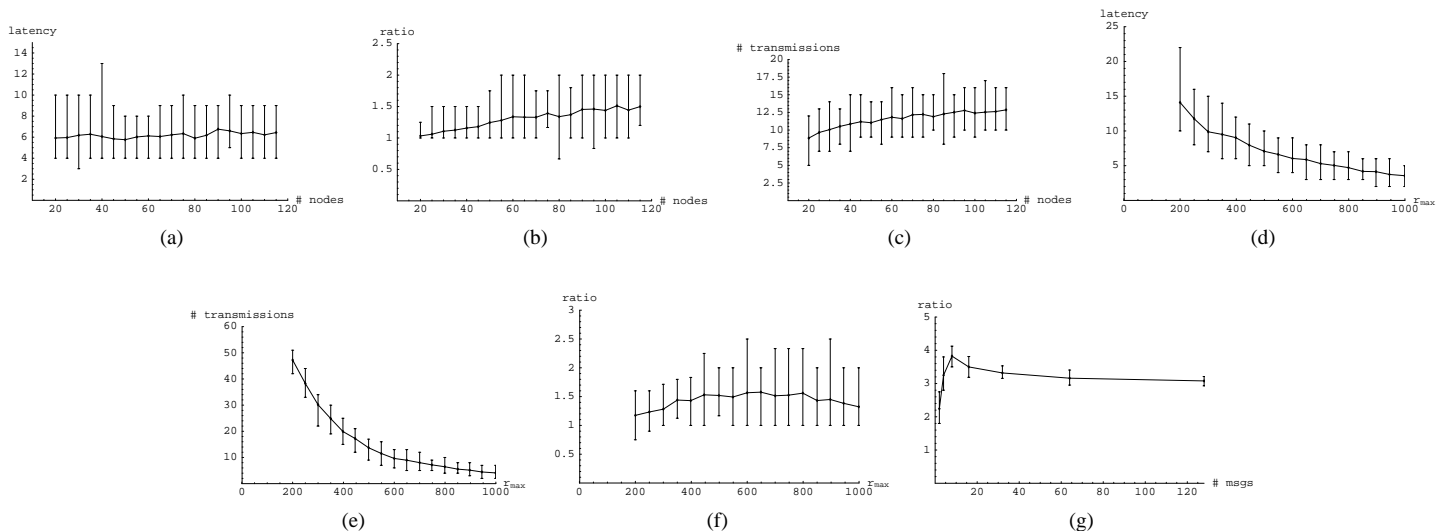
The main goal of our simulations is to compare the theoretical and experimental performance of our algorithms. To this effect, we assume that nodes possess local broadcast functionality at the MAC layer and do not employ packet level simulations. Our experiments focus on the effect of various network conditions on the performance metrics of interest.

In our experiments, the network nodes were placed uniformly at random within a square. All the experiments were performed on strongly connected graphs. All data points were averaged over 50 simulation runs and the error bars indicate the maximum and minimum deviation of any observed value from the mean, i.e, a confidence of 100% that the mean is within the range indicated by the error bars. The choice of parameters made in our experiments were inspired in part by the survey by Williams and Camp [27] and by the need to complete the simulations within reasonable time. Our simulations were performed using Mathematica. The plots referenced below refer to the plots in Figure 7.

### A. Single Message Broadcast

1) *Uniform Transmission Ranges:* We present the results of our study of the single message broadcast algorithm with uniform node transmission range. In these experiments, we placed the nodes uniformly at random in a square of length 350 meters and chose the transmission range to be 100 meters. We varied the number of nodes from 20 to 110 in steps of 5, leading to higher node densities and higher average node degree. We studied the effect of this on the latency and the number of transmissions. One node was chosen uniformly at random to be the source. We now present our results.

Plot (a) is of broadcast latency vs. the number of network nodes. Broadcast latency is the maximum time taken by any



**Figure 7.** Results of simulations that study the effects of various network conditions on the performance metrics of interest.

node to receive the message. The plot indicates that latency does not vary much with the number of nodes in the graph. This is intuitive, since the latency is mostly influenced by the depth of the BFS tree from the source. This depth does not depend much upon the number of nodes, but only on the length of the square and the transmission range.

Plot (b) is of the approximation ratio vs. the number of network nodes, where ratio is  $(latency/BFSDepth)$ . In reality, the plotted ratio is a conservative estimate of the real approximation ratio. As is seen from the plot, even this ratio is very close to 1 most of the times. In fact, the worst case ratio observed for all the runs is no worse than 1.75.

Plot (c) is of the no. of transmitting nodes vs. the no. of network nodes. While the number of transmissions seems to increase steadily initially, it stabilizes once the network size grows beyond 70 nodes. On an average 11 nodes transmit while the worst case among all the runs is 15 transmission.

2) *Non-uniform Transmission Ranges:* We now discuss the results of the single message broadcast algorithm with non-uniform transmission ranges. In these experiments, we placed 200 nodes in a square of length 1500 meters. The transmission ranges were chosen uniformly at random from the interval  $[r_{min}, r_{max}]$ , where  $r_{min} = 200$  meters for all experiments, and  $r_{max}$  was varied from 200 to 1000 in steps of 50.

Consider the plots latency vs.  $r_{max}$  and the number of transmissions vs.  $r_{max}$  (Plots (d) and (e)). Note that both these values drop down with increasing  $r_{max}$  values. Intuitively, as  $r_{max}$  increases, the average range of the nodes also increases. This leads to increasing number of nodes being covered for each transmission leading to a reduction in the total number of transmissions.

Note that even though the latency drops down with  $r_{max}$ , in Plot (f) the ratio  $(Latency/BFSDepth)$  stabilizes around 1.5. This is in contrast with the analytically predicted approximation ratio of  $(r_{max}^2/r_{min}^2)$ . We interpret these results as an indication of a much better average case performance than the guaranteed worst case performance.

## B. Multiple Messages

We now present the results of performance studies of the MSB algorithm. In these experiments, 100 nodes were placed in a square area of length 350 meters. All nodes had a uniform transmission range of 100 meters. The number of messages were doubled in each step from 2 to 128. All the messages originate at the same time. We studied the approximation ratio of the algorithm. The lower bound used in computation of the ratio is  $\max\{BFSDepth, \#messages\}$ , since both the  $BFSDepth$  from any source and the number of messages are lower bounds for latency.

Consider the plot of approximation ratio vs. lower bound (Plot (g)). Note how the ratio increases initially and then drops down to a constant value as the number of messages increase. In particular, the ratio reaches the peak when the number of messages is around 10. This can be explained by observing that the  $BFSDepth$  is a very poor lower bound when multiple messages are transmitted. Once the number of messages exceeds the  $BFSDepth$ , which happens at around 10, the lower bound is strengthened leading to a better approximation ratio. This plot also implies that for every additional message, the marginal increase in latency is only 4 time units.

In the case of the intermediate source broadcast (ISB), recall that expression (6) provides a tight lower bound of  $3k$  for the broadcast of multiple messages. This indicates that when the number of messages  $m$  is large, ISB will have a marginal latency of about 200 times units for every additional message. In contrast, MSB performs much better with a marginal latency of only 4 for every additional message. However, while ISB can be implemented with three DFS and one BFS traversal which is independent of the number of messages, MSB requires potentially  $2m$  DFS traversals. This is precisely the trade off between MSB and ISB referred to in Section V-C.

**Acknowledgments.** We thank S. Khuller, S. Pemmaraju, A. Srinivasan and Y-C. Wan for useful discussions. Special thanks to S. Pemmaraju for helping us with Mathematica and Combinatorica. We would also like to thank S. Pemmaraju, A.

Penttinen, and an anonymous referee for pointing out to us a mistake in our earlier version of BROADCASTTREE algorithm.

## REFERENCES

- [1] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Proc. of the 5th Ann. ACM/IEEE Int. Conf. on Mobile Comp. and Net.*, 1999, pp. 151–162.
- [2] C. Ho, K. Obraczka, G. Tsudik, and K. Viswanath, "Flooding for reliable multicast in multi-hop ad hoc networks," in *Proc. of the Int. Work. on Disc. Alg. and Meth. for Mobile Comp. and Comm.*, 1999, pp. 64–71.
- [3] J. Jetcheva, Y. Hu, D. Maltz, and D. Johnson, "A simple protocol for multicast and broadcast in mobile ad hoc networks," July 2001.
- [4] R. Sivakumar, B. Das, and V. Bharghavan, "Spine routing in ad hoc networks," *ACM/Baltzer Clus. Comp. J. (sp. issue on Mob. Comp)*, 1998.
- [5] B. Das, R. Sivakumar, , and V. Bharghavan, "Routing in ad-hoc networks using a virtual backbone," in *Proc. of the 6th Int. Conf. on Comp. Comm. and Net. (IC3N)*, 1997, pp. 1–20.
- [6] B. Das and V. Bharghavan, "Routing in ad-hoc networks using minimum connected dominating sets," in *Proceedings of the IEEE International Conference on Communications (ICC)*, vol. 1, 1997, pp. 376–380.
- [7] D. Lichtenstein, "Planar formulae and their uses," *SIAM Journal on Computing*, vol. 11, pp. 329–343, 1982.
- [8] B. Clark, C. Colbourn, and D. Johnson, "Unit disk graphs," *Discrete Mathematics*, vol. 86, pp. 165–177, 1990.
- [9] P.-J. Wan, K. Alzoubi, and O. Frieder, "Distributed construction of connected dominating set in wireless ad hoc networks," in *IEEE INFOCOM*, 2002.
- [10] X. Cheng, X. Huang, D. Li, and D.-Z. Du, "Polynomial-time approximation scheme for minimum connected dominating set in ad hoc wireless networks," *Networks*, vol. 42, no. 4, pp. 202–208, 2003.
- [11] M. V. Marathe, H. Breu, H. B. Hunt III, S. S. Ravi, and D. J. Rosenkrantz, "Simple heuristics for unit disk graphs," *Networks*, vol. 25, pp. 59–68, 1995.
- [12] H. Lim and C. Kim, "Multicast tree construction and flooding in wireless ad hoc networks," in *Proc. of the 3rd ACM Int. Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Sys*, 2000, pp. 61–68.
- [13] W. Peng and X.-C. Lu, "On the reduction of broadcast redundancy in mobile adhoc networks," in *Proceedings of First Annual Workshop on Mobile Ad Hoc Networking Computing. MobiHOC*, Aug. 11, 2000.
- [14] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint relaying: An efficient technique for flooding in mobile wireless networks," INRIA, Tech. Rep. Research Report RR-3898, Feb. 2000.
- [15] W. Peng and X. Lu, "Ahbp: An efficient broadcast protocol for mobile adhoc networks," *J. of Science and Tech. - Beijing, China*, 2000.
- [16] J. Sucec and I. Marsic, "An efficient distributed network-wide broadcast algorithm for mobile adhoc networks," Rutgers U., Tech. Rep., 2000.
- [17] W. Peng and X. Lu, "Efficient broadcast in mobile ad hoc networks using connected dominating sets," *J. of Software - Beijing, China*, 1999.
- [18] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks," *IEEE Trans. on Parallel and Distributed Systems*, pp. 14–25, Jan 2002.
- [19] L. Orecchia, A. Panconesi, C. Petrioli, and A. Vitaletti, "Localized techniques for broadcasting in wireless sensor networks," in *DIALM-POMC '04: Proceedings of the 2004 joint workshop on Foundations of mobile computing*, New York, NY, USA, 2004, pp. 41–51.
- [20] H. F. Salama, D. S. Reeves, and Y. Viniotis, "The delay-constrained minimum spanning tree problem," in *Second IEEE Symp. on Comp. and Comm.* IEEE Computer Society, 1997, pp. 699–704.
- [21] I. Chlamtac and S. Kutten, "On broadcasting in radio networks - problem analysis and protocol design," *IEEE Trans. on Comm.*, vol. 33, no. 12, pp. 1240–1246, 1985.
- [22] —, "Tree-based broadcasting in multihop radio networks," *IEEE Trans. on Comp.*, vol. 36, no. 10, pp. 1209–1223, 1987.
- [23] S. Basagni, I. Chlamtac, and D. Bruschi, "A mobility-transparent deterministic broadcast mechanism for ad hoc networks," *IEEE/ACM Trans. on Networking (TON)*, vol. 7, no. 6, pp. 799–807, 1999.
- [24] I. Chlamtac and A. Farago, "Making transmission schedule immune to topology changes in multi-hop packet radio networks," *IEEE/ACM Transactions on Networking*, pp. 23–29, 1994.
- [25] B. Chlebus, L. Gasieniec, A. Gibbons, A. Pelc, and W. Rytter, "Deterministic broadcasting in unknown radio networks," *Dist. Comp*, pp. 27–38, 2002.
- [26] P.-K. Hung, J.-P. Sheu, and C.-S. Hsu, "Scheduling of broadcasts in multihop wireless networks," in *European Wireless*, 2002.
- [27] B. Williams and T. Camp, "Comparison of broadcasting techniques for mobile ad hoc networks," in *Proc. of the 3rd ACM Int. Symp. on Mobile Ad hoc Net. and Comp.*, 2002, pp. 194–205.
- [28] R. Bar-Yehuda, O. Goldreich, and A. Itai, "On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization," *Journal of Computer and System Sciences*, vol. 45, pp. 104–126, 1992.
- [29] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg, "A lower bound for radio broadcast," *J. of Comp. and Sys. Sci.*, vol. 43, no. 2, pp. 290–298, 1991.
- [30] D. Bruschi and M. D. Pinto, "Lower bounds for the broadcast problem in mobile radio networks," *Distributed Computing*, pp. 129–135, 1997.
- [31] D. R. Kowalski and A. Pelc, "Deterministic broadcasting time in radio networks of unknown topology," in *43rd Annual Symposium on Foundations of Computer Science (FOCS)*, 2002, pp. 63–72.

**Rajiv Gandhi** Rajiv Gandhi is an assistant professor at Rutgers University-Camden. He received his Ph.D. in Computer Science in 2003 from the University of Maryland, College Park. Before starting his Ph.D., he worked as a software engineer at Qualcomm Inc. His interests lie in approximation algorithms for NP-hard problems arising in the domain of scheduling, wireless networks and clustering.

**Arunesh Mishra** Arunesh Mishra received his PhD in Computer Science from the University of Maryland, College Park in 2005. He is currently a Postdoc Research Scientist at the University of Wisconsin, Madison. His research interests include systems, wireless networking and security. He has published at various top IEEE and ACM conferences including Infocom, Sigmetrics and Mobicom.

**Srinivasan Parthasarathy** Srinivasan Parthasarathy is a research staff member at IBM T. J. Watson Research Center, Hawthorne, NY. He received his Ph.D. from the Department of Computer Science, at University of Maryland, College Park in 2006. His interests lie in algorithm design and optimization, and their applications to networking and information retrieval. His publications span several top journals and conferences including JACM, FOCS, SODA, and SIGMETRICS.