Multisource Broadcast in Wireless Networks

Scott C.-H. Huang, Hsiao-Chun Wu, *Senior Member*, *IEEE*, and Sundaraja Sitharama Iyengar, *Fellow*, *IEEE*

Abstract—Nowadays, there is urgent demand for wireless sensor network applications. In these applications, usually a base station is responsible for monitoring the entire network and collecting information. If emergency happens, it will propagate such information to all other nodes. However, quite often the message source is not a fixed node, since there may be base stations in charge of different regions or events. Therefore, how to propagate information efficiently when message sources vary from time to time is a challenging issue. None of conventional broadcast algorithms can deal with this case efficiently, since the change of message source incurs a huge implementation cost of rebuilding a broadcast tree. To deal with this difficult problem, we make endeavor in studying multiple source broadcast, in which targeted algorithms should be source-independent to serve the practical need. In this paper, we formulate the *Minimum-Latency Multisource Broadcast* problem. We propose a novel solution using a fixed shared backbone, which is independent of the message sources and can be used repeatedly to reduce the broadcast latency. To the best of our knowledge, our work is deemed the first attempt to design such a multisource broadcast algorithm with a derived theoretical latency upper bound.

Index Terms—All-to-all broadcast, wireless networks, distributed algorithms.

1 Introduction

We studied the *Minimum-Latency Multisource Broadcast* in wireless networks. Our objective is to minimize the maximum latency of broadcasting a message from a subset of nodes (called the *source subset*) to all other nodes in the network. Moreover, the source subset may vary over time. Although there already exist several algorithms in the literature to minimize the single-source broadcast latency in a network [1], [2], these algorithms were designed to propagate information on a routing tree rooted at a fixed source. The routing tree depends on the source. If the source changes, the routing tree needs to be constructed again. As a result, none of the existing algorithms can deal with the scenario that the source changes over time without incurring a prohibitively high memory requirement or routing-tree construction cost.

The main contribution of this work is that we design a broadcast algorithm *independent of* the source subset. We construct a *shared backbone* (c.f. routing tree) for all possible source subsets, so this shared backbone only needs to be constructed once. Whenever one or more nodes need to broadcast a message to the entire network, we can use the same shared backbone repeatedly. Thus, the multisource broadcast can be dealt with in practice. To the best of our knowledge, our work is the first attempt to design a multisource broadcast algorithm with a proven theoretical latency upper bound.

- S.C.-H. Huang is with the Department of Electrical Engineering, National Tsing Hua University, Hsinchu 30013, Taiwan, ROC. E-mail: chhuang@ee.nthu.edu.tw.
- H.-C. Wu is with the Department of Electrical and Computer Engineering, Louisiana State University, Baton Rouge, LA 70803. E-mail: wu@ece.lsu.edu.
- S.S. Iyengar is with the Department of Computer Science, Louisiana State University, Baton Rouge, LA 70803. E-mail: iyengar@csc.lsu.edu.

Manuscript received 4 Nov. 2010; revised 3 Nov. 2011; accepted 14 Dec. 2011; published online 30 Dec. 2011.
Recommended for acceptance by X. Cheng.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2010-11-0659.
Digital Object Identifier no. 10.1109/TPDS.2011.310.

The rest of this paper is outlined as follows: We present the preliminaries in Section 2. Our main multisource broadcast algorithm (Algorithm 6) and its latency upper bound (Theorem 1) will be presented in Section 3. We introduce a heuristic multisource broadcast algorithm (Algorithm 10) in Section 4 which does not have a proven theoretical latency bound but the simulation shows that this algorithm actually achieves a much lower latency than Algorithm 6. Related work is presented in Section 5. Conclusion and future work will be stated in Section 6. A supplementary document to this paper contains the appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.310. In the appendix, available in the online supplemental material, we first present two very important techniques called *tessellation* and *coloring* for the design of our algorithms. Some examples for algorithmic illustration as well as numerical results are also presented in the appendix, available in the online supplemental material. Moreover, all proofs of our theorems and lemmas are also provided in the appendix, available in the online supplemental material.

2 Preliminaries

A wireless network is modeled as a *unit disk graph* (UDG), $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$, where $\mathcal{V}$ and $\mathcal{E}$ are the node and edge sets, respectively. Throughout this work, we assume that $\mathcal{G}$ is connected. Two nodes u and v are *adjacent* in $\mathcal{G}$ if and only if their euclidean distance in between is less than 1. Time is assumed to be discrete and synchronized across the network by a global clock. Each node is able to read a variable, denoted by "$Time$," representing its clock value. Message transmissions are allotted into the synchronized time slots of equal length. In each time slot, a node can either transmit or receive a message but cannot carry both out simultaneously. Due to the broadcast nature of a wireless channel, whenever a node transmits a message, all its neighbors are aware of

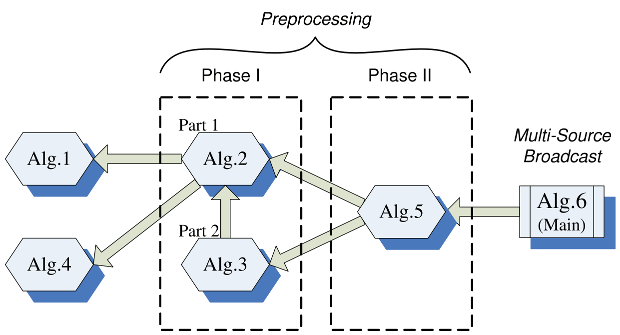


Fig. 1. Dependence relations of our algorithms.

this transmission. If two or more neighbors around a node w transmit in the same time slot, w will not successfully receive any message; in this case, we also say that collision occurs at the node w. Throughout this work, we also assume that each node in the network has a unique ID.

2.1 Problem Formulation

In a network represented as a UDG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a *single-source broadcast schedule* with respect to a fixed source node $s \in \mathcal{V}$ can be represented as a function f_s as follows: $f_s : \mathcal{V} \times \mathbb{N} \to \{0, 1\}$, where $f_s(v, t) = 1$ if $v \in \mathcal{V}$ transmits in time slot $t \in \mathbb{N}$. Throughout this paper, we say that a node v is scheduled to transmit in time slot t if and only if we set $f_s(v, t) = 1$. A node $u \in \mathcal{V}$ is said to *receive successfully* or *receive collision-free* in time slot t from a neighbor $v \in \mathcal{V}$ if and only if $f_s(v, t) = 1$ and $f_s(w, t) = 0$, $\forall w \neq v$ such that w is a neighbor of u. If a node v is scheduled to transmit a message in time slot t, it is required that either v is the source or v has received the message successfully from a relay node in an earlier time slot. It is also required that each node in $\mathcal{V}$ except for the source s will eventually receive the message successfully (either from s directly or through a relay node). Given the source s, the *latency* of a single-source broadcast schedule f_s is the last time slot such that there are still some node(s) transmitting. Formally, the latency of f_s can be defined as $lat(f_s) \overset{\text{def}}{=} \max\{t \in \mathbb{N} | f_s(v, t) = 1, v \in \mathcal{V}\}$. A *multi-source broadcast schedule* with respect to a fixed nonempty source subset $\mathcal{U} \subset \mathcal{V}$ can be represented as a function $f_{\mathcal{U}} : \mathcal{V} \times \mathbb{N} \to \{0, 1\}$, in which $f_{\mathcal{U}}$ depends on $\mathcal{U}$ instead of a single node. Each source node $u \in \mathcal{U}$ is preloaded with a source-dependent message m_u, and the objective is to distribute m_u to $\mathcal{V} - \{u\}$, for each $u \in \mathcal{U}$. The latency of $f_{\mathcal{U}}$ is defined as $lat(f_{\mathcal{U}}) \overset{\text{def}}{=} \max\{t \in \mathbb{N} | f_{\mathcal{U}}(v, t) = 1, v \in \mathcal{V}\}$, the *optimal latency* (with respect to $\mathcal{U}$ alone) is defined as $opt_{\mathcal{U}} \overset{\text{def}}{=} \min_f lat(f_{\mathcal{U}})$, and the *competitive ratio*[1] (with respect to f and $\mathcal{U}$) is defined as $\mathrm{cr}(f_{\mathcal{U}}) \overset{\text{def}}{=} \frac{lat(f_{\mathcal{U}})}{opt_{\mathcal{U}}}$.

Let $\mathcal{S}$ denote the set of all broadcast schedules for $\mathcal{G}$. A *multisource broadcast schedule family* $\mathcal{F}$ is defined as a mapping from the power set of $\mathcal{V}$ to $\mathcal{S}$. Formally, $\mathcal{F} : 2^{\mathcal{V}} \to \mathcal{S}$. Following this definition, $\mathcal{F}(\mathcal{U})$ is a multisource broadcast schedule for $\mathcal{U}$. The *worst cast competitive ratio* for the multisource broadcast schedule family $\mathcal{F}$ is defined as $\mathrm{wcr}(\mathcal{F}) \overset{\text{def}}{=} \max_{\mathcal{U} \subset \mathcal{V}} \mathrm{cr}(\mathcal{F}(\mathcal{U}))$. The *minimum-latency multi-source broadcast* problem can be defined as follows: Given a UDG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, find a multisource broadcast schedule family $\mathcal{F}$ such that $\mathrm{wcr}(\mathcal{F}(\mathcal{U}))$ is minimized. We also assume that a node can aggregate[2] (combine) multiple messages into a single one for later relay to save bandwidth.

3 Our Proposed Multisource Broadcast Algorithm

Here, in this section, we propose our main algorithm to tackle the multisource broadcast problem. The main algorithm of this section is Algorithm 6. In order to present this algorithm clearly, we need to introduce many other relevant algorithms as the required subroutines.

Our multisource broadcast algorithm is Algorithm 6, and before executing it we need the preprocessing. The preprocessing stage involves two phases: 1) *Shared backbone construction* and 2) *Distributed handshaking*. Phase 1 is undertaken in a centralized fashion, but it needs to be done only once and can be performed offline. Since Phase 1 involves a very lengthy algorithm, we break it into Part 1 (Algorithm 2) and Part 2 (Algorithm 3). The core of Phase 2 is Algorithm 5, which is fully distributed. In order to facilitate this algorithm, Phase 1 needs to be carried out as a precondition. In Algorithm 2, we run Algorithms 1 and 4 as the necessary subroutines, so we need to introduce Algorithms 1-5 before presenting Algorithm 6 in this section. The dependence relations among these algorithms are illustrated in Fig. 1.

3.1 Phase 1: Shared Backbone Construction

Here, we present the method of constructing a shared backbone. The detailed steps are presented in Algorithms 1, 2, and 3. Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we first run Algorithm 1 to obtain a subset of nodes $\mathcal{P}$, called the set of *primary nodes*.[3] Then, we run Algorithm 2 followed by Algorithm 3

1. Although the competitive ratio is indeed a main metric for broadcast scheduling algorithms, due to the design of our algorithms, the overall latencies of our scheduling algorithms actually do not depend on $\mathcal{U}$. We will establish an upper bound for the competitive ratio.

2. Data aggregation means two data packets can be merged into one single packet of the same size. This assumption is practical in most cases. For example, if the minimum or maximum value of two packets are considered, only one of them is needed for further consideration and the other can be therefore discarded. Moreover, if each packets do not carry a very big amount of information (such as video packets), two packets can be aggregated in most cases since there is often redundancy in the design of these data packets.

3. Since Algorithm 1 is deterministic, its output set can be regarded as the definition of primary nodes.

to obtain a set of *secondary nodes* or *connectors*, denoted by $\mathcal{S}$, and then we add edges to result in a connected shared backbone $\mathcal{H} = (\mathcal{V}, \mathcal{E}_{\mathcal{H}})$. Note that $\mathcal{H}$ possesses the important properties stated in Lemma 5 in the appendix, available in the online supplemental material. The remaining nodes not chosen as either primary or secondary nodes are referred to as the *tertiary nodes*.

Algorithm 1. Primary Node Selection
Input: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a node ordering $\mathcal{O} = (v_1, v_2, \ldots, v_n)$
Output: A subset $\mathcal{P} \subset \mathcal{V}$.
$\mathcal{P} \leftarrow v_1$.
for $j \leftarrow 2$ to n **do**
    Add v_j to $\mathcal{P}$ if v_j is not adjacent (with respect to $\mathcal{G}$) to any node in $\mathcal{P}$.
end for
return $\mathcal{P}$.

Algorithm 2. Shared Backbone Construction (Part I)
Precondition: A connected unit disk graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Postcondition: This algorithm outputs a subgraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}_{\mathcal{H}})$ of $\mathcal{G}$ and an auxiliary graph $\mathcal{J} = (\mathcal{V}_{\mathcal{J}}, \mathcal{E}_{\mathcal{J}})$ possessing the five properties stated in Lemma 5.
Apply Alg. 1 with an arbitrary node ordering of $\mathcal{V}$ to obtain $\mathcal{P}$.
$\mathcal{S} \leftarrow \emptyset$. $\mathcal{V}_{\mathcal{J}} \leftarrow \mathcal{P}$. $\mathcal{E}_{\mathcal{J}} \leftarrow \emptyset$.
for each pair $\{u, v\} \subseteq \mathcal{P}$ such that u, v are within 3 hops **do**
    if there exist node(s) that is adjacent to both u and v **then**
        Pick the one with the largest ID and denote it by w_0. We call w_0 the *sole* connector for u and v.
        Add w_0 to $\mathcal{S}$. Add (u, w_0) and (v, w_0) to $\mathcal{E}_{\mathcal{H}}$.
        Add (u, v) to $\mathcal{E}_{\mathcal{J}}$.
    else
        Compare the IDs of u and v. Without loss of generality, we may assume that u is the one with the larger ID.
        Consider each pair of nodes (w_u, w_v) such that w_u is adjacent to u, w_v is adjacent to v, and w_u, w_v are adjacent to each other.
        From these w_u's, we pick the one with the largest ID and denote it by w_1.
        We then look at these w_v's such that w_v is also a neighbor of w_1, and choose the one with the largest ID to be denoted by w_2.
        We call w_1 and w_2 the *first-hop* and *second-hop* connectors for u and v, respectively.
        Add w_1, w_2 to $\mathcal{S}$, and add (u, w_1), (w_2, v), and (w_1, w_2) to $\mathcal{E}_{\mathcal{H}}$.
        Add (u, v) to $\mathcal{E}_{\mathcal{J}}$.
    end if
end for
Run Alg. 4 to obtain label and parent information.
For each $x \in \mathcal{V} - \mathcal{P}$, add $(x, \mathrm{pr}(x))$ to $\mathcal{E}_{\mathcal{H}}$. /* Note that $\mathrm{pr}(x)$ is an output of Alg. 4. */

Algorithm 3. Shared Backbone Construction (Part II: Information Management)
Precondition: Same as Alg. 2 (Alg. 2 has been executed as well).
Postcondition: The three properties stated in Lemma 6.
/* Update topology information */
Each primary node $u \in \mathcal{P}$ saves its local topology information as follows: For each neighbor v in $\mathcal{J}$, u saves the corresponding sole connector or the first-hop/second-hop connector pair.
Each secondary node in $\mathcal{S}$ locally saves all of its 1-hop or 2-hop primary neighbors as well as whether it is a sole, first-hop, or second-hop connector for any pair of them.
/* Update coloring information */
Apply the hexagonal coloring method and obtain the C_{12} and C_{37} colorings.
Each primary node $u \in \mathcal{P}$ locally saves its own colors $C_{12}(u), C_{37}(u)$.
Each secondary node in $\mathcal{S}$ locally saves the coloring information $C_{12}(u), C_{37}(u)$ for each primary node u within 2 hops.
/* Update rank information */
for each $u \in \mathcal{P}$ **do**
    Sort all neighbors of u in $\mathcal{J}$ to form a list $\mathcal{L}(u)$ according to their IDs in descending order.
    Suppose $\mathcal{L}(u) = \{v_1, v_2, \ldots\}$, in which v_1 is the one with the largest ID. Define the rank function $\mathrm{rk}(u, v_j)$ as follows: $\mathrm{rk}(u, v_j) \stackrel{\text{def}}{=} j$. u locally saves the rank information $\mathrm{rk}(u, v)$ for each neighbor v in $\mathcal{J}$.
end for
Each node in $\mathcal{S}$ locally saves the rank information $\mathrm{rk}(u, v)$ for each pair of primary nodes u, v within 2 hops in $\mathcal{G}$.
/* Update label information */
Each node $z \notin \mathcal{P}$ locally saves the label information $\mathrm{lb}(z)$ and the parent information $\mathrm{pr}(z)$. The corresponding primary node $\mathrm{pr}(z)$ also saves the information about z and $\mathrm{lb}(z)$.
Let $\Delta_{\max} \stackrel{\text{def}}{=} \max_{u \in \mathcal{P}} |\{z | \mathrm{pr}(z) = u\}|$. Each primary and secondary node locally saves $\Delta_{\max}$.
Let $\mathrm{R}_{\max} \stackrel{\text{def}}{=}$ the maximum hop distance in $\mathcal{J}$ between any pair of nodes $u, v \in \mathcal{P}$. Each primary and secondary node locally saves $\mathrm{R}_{\max}$.

Algorithm 4. Label Finder
Precondition: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Alg. 2 has been run.
Postcondition: $\mathrm{lb}(z)$ and $\mathrm{pr}(z)$ will be well-defined for each $z \in \mathcal{V} - \mathcal{P}$.
Initialize $i \leftarrow 1, \mathcal{X}_0 \leftarrow \mathcal{P}, \mathcal{Z} \leftarrow \mathcal{V} - \mathcal{P}$
while $\mathcal{Z} \neq \emptyset$ **do**
    Initialize $\mathcal{W} \leftarrow \mathcal{X}_{i-1}$.
    for each $x \in \mathcal{X}_{i-1}$ **do**
        if each element in $\mathcal{Z}$ is adjacent to at least one element in $\mathcal{W} - \{x\}$, then we remove x from $\mathcal{W}$.
    end for
    $\mathcal{X}_i \leftarrow \mathcal{W}$
    for each $u \in \mathcal{X}_i$ **do**
        Find a neighbor $z \in \mathcal{Z}$ of u such that z is not

adjacent to any other node in $\mathcal{X}_i$. Set $\mathrm{lb}(z) \leftarrow i$, $\mathrm{pr}(z) \leftarrow u$, $\mathcal{Z} \leftarrow \mathcal{Z} - \{z\}$
    end for
    $i \leftarrow i + 1$
end while

3.2 Phase 2: Distributed Handshaking

In this phase, we introduce Algorithm 5. The correctness and time complexity of Algorithm 5 will be studied via Lemma 10 and Theorem 5 in the appendix, available in the online supplemental material, respectively. Their proofs can also be found in the appendix, available in the online supplemental material.

Algorithm 5. Distributed Handshaking Algorithm
Precondition: Algs. 2 and 3 have been run. Each primary node has a message m_u to transmit. All nodes have the same starting time T_s. This algorithm is run locally at each node x in the network.
Postcondition: Each primary node will receive the message m_v collision-free from each neighbor v in $\mathcal{J}$.

if $x \in \mathcal{P}$ **then** /* primary to first-hop */
    Schedule x to transmit m_x when $Time = T_s + C_{12}(x)$.
else if x is the sole or first-hop connector for some primary nodes u, v **then** /* first-hop to second-hop */
    x waits to receive m_u from u until $Time = T_s + 12$.
    Schedule x to relay m_u when $Time = T_s + 12 + (C_{37}(u) - 1)*40 + \mathrm{rk}(u, v)$.
else if x is the second-hop connector for some primary nodes u, v **then** /* second-hop to primary */
    x waits to receive m_u from the first-hop node until $Time = T_s + 1492$.
    Schedule x to relay m_u when $Time = T_s + 1492 + (C_{12}(v) - 1)*40 + \mathrm{rk}(v, u)$.
end if

3.3 Multisource Broadcast

Now, we introduce Algorithm 6, the main algorithm of this work. Having constructed the shared backbone, we can carry out multisource broadcast any time for any source subset. The correctness of Algorithm 6 will be analyzed using Lemma 11 in the appendix, available in the online supplemental material. Theorem 1 addresses the time complexity of Algorithm 6.

Algorithm 6. Distributed Multisource Broadcast Algorithm
Precondition: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a source subset $\mathcal{U} \subset \mathcal{V}$ is given. Each node $s \in \mathcal{U}$ has a message m_s to transmit to the entire network. Algs. 2 and 3 have been run. Starting time is 0. This algorithm is run locally at each node x in the network.
Postcondition: Each node in the network will receive m_s collision-free for each $s \in \mathcal{U}$.

1: **if** $x \in \mathcal{U} \setminus \mathcal{P}$ **then** Schedule s to transmit m_s to $\mathrm{pr}(s)$ at time $\mathrm{lb}(s)$.
2: **end if**
3: **if** $x \in \mathcal{U} \cap \mathcal{P}$ **then**
4:     Wait until $Time = \Delta_{\max}$
5:     u merges its message m_u with all of its received messages, if any, into a packet m'_u.
6:     Append number 1 at the end of the message m'_u, so the message becomes $\{\mathrm{m}'_u, 1\}$. Apply Alg. 5 with this message at $Time = \Delta_{\max} + 1$.
7:     **repeat**
8:         Keep receiving message(s) from its neighbors.
9:         Upon receipt of a message $\{\mathrm{m}, k\}$, check whether $k \leq \mathrm{R}_{\max}$. If yes, relay the message $\{\mathrm{m}, k+1\}$ when $Time \equiv \Delta_{\max} \bmod 1972$ by applying Alg. 5.
10:     **until** $Time = \Delta_{\max} + 1972 * \mathrm{R}_{\max}$
11:     x merges all of its received messages into a single message and then transmits at $Time = \Delta_{\max} + 1972 * \mathrm{R}_{\max} + C_{12}(x)$.
12: **end if**
13: **if** $x \in \mathcal{P} \setminus \mathcal{U}$ **then**
14:     **repeat**
15:         Keep receiving message(s) from its neighbors.
16:         Upon receipt of a message $\{\mathrm{m}, k\}$, check whether $k \leq \mathrm{R}_{\max}$. If yes, relay the message $\{\mathrm{m}, k+1\}$ when $Time \equiv \Delta_{\max} \bmod 1972$ by applying Alg. 5.
17:     **until** $Time = \Delta_{\max} + 1972 * \mathrm{R}_{\max}$
18:     x merges all of its received messages into a single message and then transmits at $Time = \Delta_{\max} + 1972 * \mathrm{R}_{\max} + C_{12}(x)$.
19: **end if**

Theorem 1. *The time complexity of our multisource broadcast algorithm (Algorithm 6) is* $\Delta_{\max} + 1972 * \mathrm{R}_{\max} + 12$ *time slots.*

Since the time complexity can be obtained directly from Line 18 of Algorithm 6, we omit the proof.

4 Two-Separated Multisource Broadcast Algorithm

In this section, we want to show a modified version of Algorithm 6 by constructing a different shared backbone having the *2-separation property* (see the postcondition of Lemma 12). This algorithm will significantly reduce the time complexity of the handshaking algorithm (Algorithm 5) and therefore may significantly reduce the multisource broadcast latency. However, such a property arises at a cost of potentially increasing $\mathrm{R}_{\max}$ and therefore increasing the broadcast latency. Theoretically speaking, this increase can be large as there can exist a certain topology such that constructing a 2-separated backbone could significantly increase $\mathrm{R}_{\max}$. However, according to our simulations, this increase is usually not very large. Therefore, this fact usually leads to a practical multisource broadcast algorithm with much less time complexity, although it is not theoretically guaranteed.

The main algorithm of this section is Algorithm 10. Similarly, in order to present this algorithm clearly, we need to introduce several other algorithms as subroutines. Our multisource broadcast algorithm is Algorithm 10, and before executing it we need to do preprocessing. The preprocessing stage involves two phases: 1) *2-separated shared backbone construction* and 2) *2-separated distributed handshaking*. Phase 1 is carried out in a centralized fashion, but it needs to be done only once and can be performed offline. Since Phase 1 is a very lengthy algorithm, we break it into Part 1 (Algorithm 7) and Part 2 (Algorithm 8). The core of Phase 2 is Algorithm 9, which is a fully distributed algorithm. In order to facilitate Phase 2, Phase 1 needs to be undertaken as a precondition. In Algorithm 7, we run Algorithm 1 as a subroutine, so we need to introduce Algorithms 7-9 before presenting Algorithm 10. The dependence relations among these algorithms are shown in Fig. 1.

We present the method of constructing a 2-separated shared backbone. The detailed steps are presented in Algorithms 7 and 8. The 2-separated shared backbone $\mathcal{H}'$ with the auxiliary graph $\mathcal{J}'$ obtained by running Algorithms 7 and 8 possesses important properties presented in Lemma 12 in the appendix, available in the online supplemental material. The correctness of Algorithm 10 will be studied in Lemma 16 in the appendix, available in the online supplemental material. Theorem 2 addresses the time complexity of Algorithm 10.

Algorithm 7. 2-Separated Backbone Construction (Part I)
Precondition: A connected unit disk graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Postcondition: This algorithm outputs a subgraph $\mathcal{H}' = (\mathcal{V}, \mathcal{E}_{\mathcal{H}'})$ of $\mathcal{G}$ and an auxiliary graph $\mathcal{J}' = (\mathcal{V}_{\mathcal{J}}, \mathcal{E}_{\mathcal{J}'})$ such that Lemma 12 holds.
Arbitrarily choose a node g in $\mathcal{V}$.
Sort all nodes in $\mathcal{V}$ according to its hop-distance to g in increasing order. Let $\mathcal{O} = (v_1, v_2, \ldots)$ denote this ordering.
Apply Alg. 1 with $\mathcal{O}$ to obtain $\mathcal{P}$.
$\mathcal{S} \leftarrow \emptyset$. $\mathcal{V}_{\mathcal{J}} \leftarrow \mathcal{P}$. $\mathcal{E}_{\mathcal{J}'} \leftarrow \emptyset$.
for each pair $\{u, v\} \subseteq \mathcal{P}$ such that u, v are within 2 hops **do**
    Look at the nodes adjacent to both u and v. Pick the one with the largest ID and denote it by w_0. We call w_0 the *sole connector* for u and v.
    Add w_0 to $\mathcal{S}$. Add (u, w_0) and (v, w_0) to $\mathcal{E}_{\mathcal{H}'}$.
    Add (u, v) to $\mathcal{E}_{\mathcal{J}'}$.
end for
Run Alg. 4 to get label and parent information.
For each $x \in \mathcal{V} - \mathcal{P}$, add $(x, \mathrm{pr}(x))$ to $\mathcal{E}_{\mathcal{H}'}$.

Algorithm 8. 2-Separated Backbone Construction (Part II: Information Management)
Precondition: Same as Alg. 7 (Alg. 7 has been executed).
Postcondition: The four properties stated in Lemma 8.
/* Update topology information */
Each primary node $u \in \mathcal{P}$ saves its local topology information as follows: For each neighbor v in $\mathcal{J}'$, u saves the corresponding connector.
Each secondary node in $\mathcal{S}$ locally saves all of its primary neighbors as well as whether it is a connector for any pair of them.
Apply the hexagonal coloring method and obtain the C_{12} coloring. /* Update coloring information */
Each primary node $u \in \mathcal{P}$ locally saves its own color $C_{12}(u)$.
Each secondary node in $\mathcal{S}$ locally saves the coloring information $C_{12}(u)$ for each primary neighbor u (in $\mathcal{G}$).
for each $u \in \mathcal{P}$ **do** /* Update rank information */
    Sort all neighbors of u in $\mathcal{J}'$ to form a list $\mathcal{L}(u)$ according to their IDs in descending order.
    Suppose $\mathcal{L}(u) = \{v_1, v_2, \ldots\}$, in which v_1 is the one with the largest ID. Define the rank function $\mathrm{rk}(u, v_j)$ as follows: $\mathrm{rk}(u, v_j) \stackrel{\text{def}}{=} j$. u locally saves the rank information $\mathrm{rk}(u, v)$ for each neighbor v in $\mathcal{J}'$.
end for
Each node in $\mathcal{S}$ locally saves the rank information $\mathrm{rk}(u, v)$ for all primary nodes u, v within 2 hops in $\mathcal{G}$.
Each node $z \notin \mathcal{P}$ locally saves the label information $\mathrm{lb}(z)$ and the parent information $\mathrm{pr}(z)$. The corresponding primary node $\mathrm{pr}(z)$ also saves the information about z and $\mathrm{lb}(z)$. /* Update label information */
Let $\Delta_{\max} \stackrel{\text{def}}{=} \max_{u \in \mathcal{P}} |\{z | \mathrm{pr}(z) = u\}|$. Each primary and secondary node locally saves $\Delta_{\max}$.
Let $\mathrm{R}_{\max} \stackrel{\text{def}}{=}$ the maximum hop distance in $\mathcal{J}'$ between any pair of nodes $u, v \in \mathcal{P}$. Each primary and secondary node locally saves $\mathrm{R}_{\max}$.

Algorithm 9. 2-Separated Distributed Handshaking Algorithm
Precondition: Algs. 7 and 8 have been executed. Each primary node has a message m_u to transmit. All nodes have the same starting time T_s. This algorithm is run locally at each node x in the network.
Postcondition: Each primary node will receive the message m_v collision-free from each neighbor v in $\mathcal{J}'$.
if $x \in \mathcal{P}$ **then** /* primary to connector */
    Schedule x to transmit m_x when $Time = T_s + C_{12}(x)$.
else if x is a connector for some primary nodes u, v **then**
    x waits to receive m_u from u until $Time = T_s + 12$. /* connector to primary */
    Schedule x to relay m_u when $Time = T_s + 12 + (C_{12}(v) - 1) * 20 + \mathrm{rk}(v, u)$.
end if

Algorithm 10. Distributed 2-Separated Multi-Source Broadcast Algorithm
Precondition: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a source subset $\mathcal{U} \subset \mathcal{V}$ is given. Each node $s \in \mathcal{U}$ has a message m_s to transmit to the entire network. Algs. 7 and 8 have been run. Starting time is 0. This algorithm is run locally at each node x in the network.
Postcondition: Each node in the network will receive m_s collision-free for each $s \in \mathcal{U}$.
if $x \in \mathcal{U} \setminus \mathcal{P}$ **then**
    Schedule s to transmit m_s to $\mathrm{pr}(s)$ at time $\mathrm{lb}(s)$.
end if
if $x \in \mathcal{U} \cap \mathcal{P}$ **then**
    Wait until $Time = \Delta_{\max}$
    u merges its message m_u with all of its received messages, if any, into a packet m'_u.
    Append number 1 at the end of the message m'_u, so the message becomes $\{\mathrm{m}'_u, 1\}$. Apply Alg. 9 with this message at $Time = \Delta_{\max} + 1$.
    repeat
        Keep receiving message(s) from its neighbors.
        Upon receipt of a message $\{\mathrm{m}, k\}$, check whether $k \leq \mathrm{R}_{\max}$. If yes, relay the message $\{\mathrm{m}, k + 1\}$ when

$Time \equiv \Delta_{\max} \bmod 252$ by applying Alg. 9.
11: **until** $Time = \Delta_{\max} + 252 * \mathrm{R}_{\max}$
12: x merges all of its received messages into a single message and transmits at
$Time = \Delta_{\max} + 252{*}\mathrm{R}_{\max} + C_{12}(x)$.
13: **end if**
14: **if** $x \in \mathcal{P} \setminus \mathcal{U}$ **then**
15: **repeat**
16: Keep receiving message(s) from its neighbors.
17: Upon receipt of a message $\{\mathrm{m}, k\}$, check whether $k \leq \mathrm{R}_{\max}$. If yes, relay the message $\{\mathrm{m}, k+1\}$ when $Time \equiv \Delta_{\max} \bmod 252$ by applying Alg. 5.
18: **until** $Time = \Delta_{\max} + 252 * \mathrm{R}_{\max}$
19: x merges all of its received messages into a single message and transmits at
$Time = \Delta_{\max} + 252 * \mathrm{R}_{\max} + C_{12}(x)$.
20: **end if**

Theorem 2. *The time complexity of our 2-separated multisource broadcast algorithm (Algorithm 10) is* $\Delta_{\max} + 252{*}\mathrm{R}_{\max} + 12$ *time slots.*

Since the time complexity can be obtained directly from Line 19 of Algorithm 10, we omit the proof.

5 Related Work

Many research works regarding the single-source broadcast scheduling problems in wireless networks have been dedicated in the literature in the last two decades. They can be classified into two categories in terms of graph models, namely general graphs and disk graphs.

In the first category, networks are modeled as arbitrary undirected graphs. Most of the existing works focus on designing the deterministic centralized scheduling algorithms [3], [4], [5], [6], [7], [8], [9]. Among the aforementioned works, the best latency result was achieved as $O(R + log^2 n)$ by Kowalski and Pelc in [9], where n is the number of nodes and R is the radius of the graph with respect to the source. On the other hand, the deterministic distributed scheduling algorithms were considered in [10], [11]. Moreover, some typical randomized algorithms of Las Vegas type were proposed in [12], [13].

The recent prevalent network model is based on the disk graph when each node has a different transmission range. It can also be further simplified as a Unit Disk Graph when the transmission ranges of the the nodes are identical. By taking advantage of the geometric property of UDG, the scheduling algorithms with constant approximation ratios were proposed in [1], [2], [14], [15], [16]. To be specific, Dessmark and Pelc in [14] presented a broadcast schedule of 2,400-approximation. Gandhi et al. in [1] proposed an approximation algorithm with a ratio around 648. Huang et al. in [2] improved this approximation ratio to 16. In addition, when the interference range is different from the transmission range for each node, Chen et al. in [15] gave a $2\pi\alpha^2$-approximation algorithm, where $\alpha > 1$ is the ratio of the interference range to the transmission range. Shang et al. in [16] further improved the approximation ratio to $(1 + 2\alpha)^2 + 32$. Basically, these existing works rely on the same idea of propagating messages along a fixed-source broadcast tree and designing the appropriate transmission schedules in order to avoid collisions.

Apart from the broadcast algorithms above, Bar-Yehuda and Israeli in [17] considered the k-point-to-point transmission problem. Lee et al. in [18] studied the multisource broadcast problem and designed a randomized algorithm. However, they did not derive any theoretical latency bound. Related problems such as beaconing, gossiping, and dominating set construction were studied in [19], [20], and [21], respectively. To the best of our knowledge, this work is the first attempt to design a multisource broadcast algorithm with a proven theoretical latency bound.

6 Conclusion and Future Work

In this paper, we study the Minimum-Latency Multisource Broadcast problem and design two broadcast algorithms (Algorithms 6 and 10) to reduce the latency. Since multisource broadcast is very time consuming and the associated latency bound cannot be found in the existing literature, our work can be deemed as the first attempt to tackle with this problem. Algorithm 6 leads to a guaranteed latency upper bound in terms of maximum hop distance, while Algorithm 10 does not have. Although Algorithm 10 does not have a theoretical latency upper bound, heuristically speaking, it can lead to a significantly lower latency and therefore it is very useful in practice. For our future work, power limitation is an important problem. We believe that we could extend this work and consider power limitation in light of [22]. Moreover, the unit disk graph model adopted in this paper could be replaced by more practical interference models such as two-disk or signal-to-interference-plus-noise ratio (SINR) models.

Acknowledgments

S. C.-H. Huang was supported by the National Science Council of Taiwan under the project 99-2218-E-007-021. H.-C. Wu was supported by Networking Technology and Systems Award (NSF-CNS 0963793) from National Science Foundation, DoD-DEPSCoR Grant (N0014-08-1-0856) from Office of Naval Research, NSF Pilot-fund, and OPT-IN grant from Louisiana Board of Regents.

References

[1] R. Gandhi, S. Parthasarathy, and A. Mishra, "Minimizing Broadcast Latency and Redundancy in Ad Hoc Networks," *Proc. MobiHoc '03*, pp. 222-232, 2003.
[2] S.C.-H. Huang, P.-J. Wan, X. Jia, H. Du, and W. Shang, "Minimum-Latency Broadcast Scheduling in Wireless Ad Hoc Networks," *Proc. IEEE INFOCOM '07*, pp. 733-739, 2007.
[3] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg, "A Lower Bound for Radio Broadcast," *J. Computer and System Sciences*, vol. 43, no. 2, pp. 290-298, 1991.
[4] M. Elkin and G. Kortsarz, "An Improved Algorithm for Radio Networks," *Proc. 16th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA '05)*, 2005.
[5] I. Gaber and Y. Mansour, "Centralized Broadcast in Multihop Radio Networks," *J. Algorithms*, vol. 46, no. 1, pp. 1-20, 2003.
[6] 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.
[7] D.R. Kowalski and A. Pelc, "Centralized Deterministic Broadcasting in Undirected Multi-Hop Radio Networks," *Proc. Int'l Workshop Approximation Algorithms for Combinatorial Optimization Problems (APPROX-RANDOM '04)*, pp. 171-182, 2004.

[8] L. Gasieniec, D. Peleg, and Q. Xin, "Faster Communication in Known Topology Radio Networks," *Proc. ACM Symp. Principles of Distributed Computing (PODC '05)*, pp. 129-137, 2005.

[9] D.R. Kowalski and A. Pelc, "Optimal Deterministic Broadcasting in Known Topology Radio Networks," *Distributed Computing*, vol. 19, pp. 185-195, 2007.

[10] D. Bruschi and M. Del Pinto, "Lower Bounds for the Broadcast Problem in Mobile Radio Networks," *Distributed Computing*, vol. 10, no. 3, pp. 129-135, 1997.

[11] M. Chrobak, L. Gasieniec, and W. Rytter, "Faster Broadcasting and Gossiping in Radio Networks," *Proc. Symp. Foundations of Computer Science (FOCS '00)*, pp. 575-581, 2000.

[12] 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," *J. Computer and System Sciences*, vol. 45, no. 1, pp. 104-126, 1992.

[13] E. Kushilevitz and Y. Mansour, "An $\Omega(D \log(N/D))$ Lower Bound for Broadcast in Radio Networks," *SIAM J. Computing*, vol. 27, pp. 702-712, 1998.

[14] A. Dessmark and A. Pelc, "Tradeoffs between Knowledge and Time of Communication in Geometric Radio Networks," *Proc. ACM Symp. Parallel Algorithms and Architectures (SPAA '01)*, pp. 59-66, 2001.

[15] Z. Chen, C. Qiao, J. Xu, and T. Lee, "A Constant Approximation Algorithm for Interference Aware Broadcast in Wireless Networks," *Proc. IEEE INFOCOM '07*, pp. 740-748, 2007.

[16] W.-P. Shang, P.-J. Wan, and X.-D. Xu, "Improved Algorithm for Broadcast Scheduling of Minimal Latency in Wireless Ad Hoc Networks," *Acta Math. Applicatae Sinica*, vol. 26, no. 1, pp. 13-22, 2010.

[17] R. Bar-Yehuda and A. Israeli, "Multiple Communication in Multi-Hop Radio Networks," *Proc. ACM Symp. Principles of Distributed Computing (PODC '89)*, pp. 329-338, 1989.

[18] C. Lee, M.H. Ammar, and J.E. Burns, "Randomized Multi-Source Broadcast Protocols in Multi-Hop Radio Networks," Technical Report: GIP-CC-93-69, Georgia Inst. of Technology, 1993.

[19] P.-J. Wan, X. Xu, L. Wang, X. Jia, and E.K. Park, "Minimum-Latency Beaconing Schedule in Multihop Wireless Networks," *Proc. IEEE INFOCOM '09*, pp. 2340-2346, 2009.

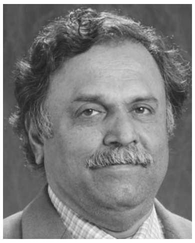
[20] S.C.-H. Huang, P.-J. Wan, H. Du, and E.K. Park, "Minimum Latency Gossiping in Radio Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 21, no. 6, pp. 790-800, June 2010.

[21] L. Gewali, K. Mohamad, and M. Tun, "Interference Aware Dominating Set for Sensor Network," *Proc. Third Int'l Conf. Information Technology: New Generations*, pp. 268-273, http://dl.acm.org/citation.cfm?id=1128011.1128119, 2006.

[22] S. Yi and Y.T. Hou, "Theoretical Results on Base Station Movement Problem for Sensor Network," *Proc. IEEE INFOCOM '08*, pp. 1-5, 2008.

Scott C.-H. Huang received the BS degree from the National Taiwan University, and the PhD degree from the University of Minnesota, Twin Cities. He is currently an assistant professor in the Department of Electrical Engineering, National Tsing Hua University. Prior to that, he had joined the faculty of Computer Sciences at City University of Hong Kong. He has published more than 30 peer-reviewed technical journal and conferences papers. His research interests include wireless ad hoc/sensor network, security, communication theory, as well as combinatorial optimization.

Hsiao-Chun Wu (M'00-SM'05) received the BSEE degree from National Cheng Kung University, Taiwan, in 1990, and the MS and PhD degrees in electrical and computer engineering from University of Florida, Gainesville, in 1993 and 1999 respectively. Since January 2001, he has joined the faculty in the Department of Electrical and Computer Engineering, Louisiana State University, Baton Rouge. He has published more than 150 peer-refereed technical journal and conference articles in electrical and computer engineering. His research interests include the areas of wireless communications and signal processing. He is an IEEE distinguished lecturer. He currently serves as an associate editor for *IEEE Transactions on Broadcasting*, *IEEE Signal Processing Letters*, *IEEE Communications Magazine*, *International Journal of Computers and Electrical Engineering*, *Journal of Information Processing Systems*, *Physical Communication*, *Journal of the Franklin Institute*, and *International Journal of Advancements in Technology*. He was a lead guest editor for *IEEE Journal of Selected Topics in Signal Processing*and *Journal of Communications*. He used to serve as an associate editor for *IEEE Transactions on Vehicular Technology*. He has also served for numerous textbooks, IEEE/ACM conferences and journals as the technical committee, symposium chair, track chair, or the reviewer in signal processing, communications, circuits and computers. He is a senior member of the IEEE.



Sundaraja Sitharama Iyengar (F'95) is currently the director and Ryder professor at Florida International University's School of Computing and Information Sciences in Miami, Florida. He was the Roy Paul Daniels chaired professor and chairman of computer science at Louisiana State University, and he is also been the chaired professor at various institutions around the world. His research interests include high-performance algorithms, data structures, sensor fusion, data mining, and Computational aspects of intelligent systems. He has coauthored eight books and edited seven books. He has published more than 380 research papers. He is a SIAM distinguished lecturer/ACM National lecturer/IEEE distinguished scientist. He has served as the editor of several IEEE journals and is the founding editor-in-chief of the *International Journal of Distributed Sensor Networks*. He has won Distinguished Research Master Award/LSU Rain Makers Award/Hub Cotton Award for faculty excellence. He is a fellow of the IEEE, ACM, AAAS, and SDPS.