An Approximation Algorithm for Conflict-Aware Broadcast Scheduling in Wireless Ad Hoc Networks

Reza Mahjourian
CISE Dept., Univ. of Florida
rezam@ufl.edu

Feng Chen
ECE Dept., Univ. of Florida
chenf@ecel.ufl.edu

Ravi Tiwari
CISE Dept., Univ. of Florida
rtiwari@cise.ufl.edu

My Thai
CISE Dept., Univ. of Florida
mythai@cise.ufl.edu

Hongqiang Zhai
Philips Research
hong.zhai@philips.com

Yuguang Fang *
ECE Dept., Univ. of Florida
fang@ecel.ufl.edu

ABSTRACT

Broadcast scheduling is a fundamental problem in wireless ad hoc networks. The objective of a broadcast schedule is to deliver a message from a given source to all other nodes in a minimum amount of time. At the same time, in order for the broadcast to proceed as predicted in the schedule, it must not contain parallel transmissions which can be conflicting based on the collision and interference parameters in the wireless network. Most existing work on this problem use a limited network model which accounts only for conflicts occurring inside the transmission ranges of the nodes. The broadcast schedules produced by these algorithms are likely to experience unpredictable delays when deployed in the network. This is because they do not take into consideration other important sources of conflict in parallel transmissions, namely the interference range and the carrier sensing range. In this paper we develop a conflict-aware network model, which uses these parameters to increase the probability of scheduling conflict-free transmissions, and thereby improve the reliability of the broadcast schedule. We present and prove correctness of a constant approximation algorithm for minimum-latency broadcast scheduling under this network model. We also present a greedy heuristic algorithm for the same problem. Experimental results are provided to evaluate the performance of our algorithms. In addition, the algorithms are analyzed to justify their performance trends.

Categories and Subject Descriptors:
C.2.1 [Computer-Communication Networks]: Network Architecture and Design - *Wireless communication*

General Terms: Algorithms, Theory

Keywords: Approximation Algorithm, Wireless Networks, Broadcast Scheduling, Conflict-Aware

*The works of Feng Chen and Yuguang Fang were partially supported by the NSF under grants CNS-0721744 and DBI-0529012.

MobiHoc'08, May 26–30, 2008, Hong Kong SAR, China.

1. INTRODUCTION

A wireless ad hoc network is a set of nodes which communicate with each other using radio transmissions. Every node in the network has a *transmission range*, which is the maximum distance where the node's signal can be correctly received. In wireless networks, only nodes which are within the transmission range of each other can communicate directly. Nodes which cannot reach each other directly must use intermediate nodes for routing their message. Ad hoc networks can operate without a centralized controller. The inherently distributed nature of the wireless nodes introduces many intriguing and challenging research problems that need to be tackled when designing applications on these networks.

One fundamental operation in wireless networks is broadcast. It is a common operation used in a variety of network applications. The objective of the broadcast operation is to deliver a message from a single source node to all the other nodes in the network. Typically, the area of a wireless ad hoc network is larger than the transmission range of any individual node. Therefore, carrying out the broadcast in general involves relaying the message by intermediate nodes. This necessitates computing a broadcast schedule. The broadcast schedule determines which nodes must transmit the message and at what times. It is computed beforehand for any network topology and then subsequently used when message broadcasts are required. The *broadcast latency* is defined to be the time by which all the nodes in the network have received the message. We are interested in producing an optimal broadcast schedule which minimizes the broadcast latency. All applications which rely on broadcasting can directly benefit from such optimization. Broadcasting is also an integral part of many distributed protocols and minimizing the time required for broadcasting can improve the performance of such protocols.

The broadcast nature of the wireless medium makes it possible to use a single transmission by any node to inform all the other nodes within its transmission range. On the other hand, this same broadcast nature can cause two parallel transmissions to fail due to collision and interference. We call such parallel transmissions to be *conflicting* with each other. A broadcast scheduling algorithm must avoid scheduling parallel transmissions which are very likely to fail based on the collision and interference parameters of the wireless network in which they operate. In this paper we study the problem of minimum-latency broadcast scheduling while taking the interference range and carrier sensing range parameters into account, so that we can avoid scheduling conflicting parallel transmissions.

The remainder of this paper is organized as follows. In Section 2 we develop the notion of conflict-aware communication and outline our contributions. Section 3 details related work. Section 4 presents

a more formal definition of the problem. A constant approximation algorithm and its analysis are discussed in Section 5. Section 6 presents the heuristic algorithm. The performance of the algorithms is evaluated experimentally through simulations in Section 7. Final conclusions are drawn in Section 8.

2. CONFLICT-AWARE COMMUNICATION

In this paper we consider the following sources of potential conflicts in parallel transmissions.

2.1 Types of Conflict in Parallel Transmissions

Two parallel transmission can be expected to succeed only if they avoid all following types of conflict:

Type 1. *Collision at Receiver:* If a node is within the transmission range of two or more transmitting nodes, it cannot correctly receive either of the messages.

Most approximation algorithms and heuristic algorithms presented in existing work on this problem [6, 9] address only this type of conflict. We denote an algorithm which avoids Type 1 conflicts to be *collision-aware*.

Type 2. *Interference at Receiver:* If a node is within the *interference range* of a transmitting node, it cannot correctly receive a message from any of its neighbors.

It is well known that wireless nodes have interference ranges larger than their transmission ranges. The interference range of a node is the maximum distance where the node's signal causes enough interference to affect the correct reception of a parallel transmission. The interference range is a parameter which is estimated from the protocol, network topology, and the environment. The only work so far to include the interference range in computing the broadcast schedule is the algorithm proposed by Chen et. al. in [1]. We denote an algorithm which avoids Type 2 conflicts to be *interference-aware*. Notice that every Type 1 conflict is also an instance of a Type 2 conflict.

Type 3. *Contention at Sender:* If a node is within the *carrier sensing range* of another transmitting node, it cannot transmit a message to its neighbors.

Distributed Medium Access Control (MAC) protocol is of great importance in coordinating medium access in wireless ad hoc networks. While there are different distributed MAC protocols for wireless networks, the widely-used MAC protocols are variations of the Carrier Sensing Multiple Access (CSMA) protocol. The ubiquitous IEEE 802.11 MAC protocol adopts CSMA with Collision Avoidance (CSMA/CA) to achieve distributed medium access control as well. In CSMA systems, a *carrier sensing range* is set in the wireless nodes to guide them on how to share the wireless medium. If a node senses another transmitter's signal inside its carrier sensing range, it is expected to refrain from occupying the channel. The ultimate goal of the CSMA protocols is to decrease the likelihood of collisions at receivers by using the carrier sensing range to regulate the transmitters. Since the CSMA protocol is widely used in wireless networks, it is useful to consider the effect of carrier sensing mechanism in our algorithm. Moreover, the broadcast scheduling algorithm is quite likely to be employed as part of a larger algorithm which might need to fine tune the value of the carrier sensing range in the network [5].

We denote an algorithm which avoids all the above types of conflicts to be *conflict-aware*

Figure 1 shows examples of transmissions which exhibit different types of conflicts. In this figure, only the transmission range (shaded region), the interference range (middle disc), and the carrier sensing range (outer disc) of nodes t_0, t_1 are shown. We denote a transmission from node t to node r by "$t \rightarrow r$". The main transmission in this figure is "$t_0 \rightarrow r_0$". All the other transmissions shown are conflicting with this main transmission. Transmission "$t_1 \rightarrow r_1$" suffers from Type 2 conflict because r_1 is within the interference range of t_0. Transmission "$t_2 \rightarrow r_2$" shows another example of Type 2 conflict because of t_2 and r_0. An example of Type 3 conflict is shown with "$t_3 \rightarrow r_3$". Finally, "$t_4 \rightarrow r_4$" exhibits all three types of conflicts with "$t_0 \rightarrow r_0$".

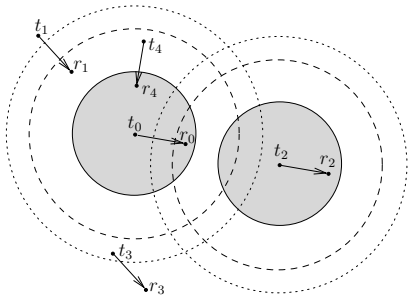

Figure 1: Examples of conflicting transmissions

Creating a completely realistic network model requires considering many more parameters and sources of failures in wireless communications, even in the absence of parallel transmissions. However, the interference range and the carrier sensing range have a considerable impact on the operation of any algorithm in wireless networks, and this is why they are included in our study.

2.2 Our Contribution

We study the problem of computing a conflict-aware minimum-latency broadcast schedule which considers all types of conflict in parallel transmissions, as introduced in Section 2.1. We provide a solution to this problem for wireless networks where the interference ranges and carrier sensing ranges of nodes are different from their transmission ranges. We assume all the nodes to have an equal transmission range r, an equal interference range αr, $\alpha \geq 1$, and an equal carrier sensing range βr, $\beta \geq 1$. We propose an approximation algorithm which is guaranteed to generate a broadcast schedule whose latency is within a constant ratio of the optimal solution. To our best knowledge, it is the only approximation algorithm presented so far which considers the interference range or the carrier sensing range. We also present variations of a heuristic algorithm which can produce near-optimal schedules. Simulation results show that these algorithm outperform existing algorithms.

3. RELATED WORK

Wireless networks are usually modeled using graphs. Different graph models have been proposed for this purpose. Commonly, vertices are used to represent nodes and edges are used to represent the reachability of nodes from one another. Some papers in the radio network literature study the problem of minimum-latency broadcast scheduling under the general graph model. However, for ad hoc networks, more restrictive graph models such as *Disk Graphs* (DGs), and *Unit Disk Graphs* (UDGs) are more suitable. Disk Graphs model wireless networks whose nodes have different

transmission ranges. Unit Disk Graphs are usually used when all nodes have equal transmission ranges. In both DGs and UDGs, there is an edge from node u to v if and only if v is located inside the transmission range of u. DGs use directed edges in the general form, but UDGs usually have undirected edges.

One of the earliest protocols suggested for broadcasting and multicasting in ad hoc networks is *flooding* [8, 10]. In the flooding protocol, every node transmits the message to all its neighbors after receiving it. Ni et al. [12] show that this protocol can lead to severe contention, collision and retransmissions. This situation is referred to as a broadcast storm.

Chlamatac and Kutten [3] study this problem in general graphs. They present an algorithm which creates a collision-aware broadcast schedule along a broadcast tree. They show that for arbitrary graphs they achieve a broadcast latency within $O(ln(n/R)^2)$ times the optimal latency, where n is the number of nodes in the network and R is the graph-theoretic radius of the network.

Gandhi, Parthasarathy, and Mishra [6] study the problem of minimum-latency broadcast in directed Disk Graphs. They create a Dominating Set in the network and then use additional secondary nodes to establish the connection between the nodes in the dominating set to produce a broadcast tree. The broadcast schedule is then formed along this broadcast tree. Their work is the first paper to prove a constant approximation ratio for minimum-latency broadcast scheduling in directed Disk Graphs. We note that their algorithm can produce disconnected broadcast trees, but fixing the problem does not affect their approximation ratio.

Scott et al. [9] present a more efficient solution to this problem for UDGs. They prove a constant approximation ratio, however they achieve a better constant compared to the work in [6]. They create the schedule based on a Connected Dominating Set in the network and broadcast the message layer by layer along this set. They use the geometric properties of UDGs to prove a lower bound of $16R - 15$. They also extend Gasieniec et al.'s pipelined algorithm [7] on arbitrary graphs to the UDG model to get a lower bound of $R + O(logR)$. This algorithm is based a standard node ranking algorithm [13] assigning ranks to the nodes on the broadcast tree. The node ranks guide the algorithm to perform the broadcast in parallel in different layers of the network.

Unfortunately, none of the above-mentioned works consider the interference range or the carrier sensing range of the wireless nodes. Chen et. al.'s paper [1] is the only work so far to study the interference ranges of the nodes. They include Type 1 and Type 2 conflicts, and present an approximation algorithm with a claimed constant of $O(\alpha^2)$, where α is the ratio of the interference range to the transmission range of the nodes. However, we could not verify the correctness of their approximation ratio because of some gaps in [1, Lemma 4] and [1, Corollary 7]. Unfortunately, we were not able to bridge these gaps.

3.1 Hardness of Minimum-Latency Broadcast Scheduling

Chlamatac and Kutten [2] prove that the problem of collision-aware minimum-latency broadcast scheduling is NP-hard in arbitrary graphs. The problem of collision-aware broadcast scheduling can be reduced to the problem of conflict-aware broadcast scheduling by setting $\alpha = 1, \beta = 1$. Therefore, conflict-aware broadcast scheduling is NP-hard in arbitrary graphs as well.

Gandhi, Parthasarathy, and Mishra [6] claim NP-hardness of collision-aware broadcast scheduling on directed Disk Graphs, although they leave out the proof due to lack of space. Chen et al. [1] show NP-hardness of interference-aware broadcast scheduling in DGs. However, to our best knowledge, none of the existing works have proved the NP-hardness of minimum-latency broadcast scheduling under the more restrictive UDG model. It is known that many NP-hard problems, including Minimum Vertex Cover, Maximum Independent Set, and Minimum Vertex Coloring remain NP-hard under the UDG model [4]. So, it is widely believed that collision-aware broadcast scheduling in UDGs is an NP-hard problem as well. However, proving NP-hardness remains an open problem. NP-hardness of conflict-aware broadcast scheduling follows from the NP-hardness of collision-aware broadcast scheduling.

4. PRELIMINARIES

In this section we first introduce some notations and definitions. We then formulate the problem as well as the network model used.

4.1 Graph-Theoretic Definitions

This section introduces some notations and definitions used in other sections. Let $G = (V, E)$ be an undirected graph. When we need to differentiate between components of different graphs, we denote the set of nodes and edges in G by $V(G), E(G)$, respectively. The subgraph of G induced by $U \subseteq V$ is denoted by $G[U]$. G is said to be connected if for any two nodes $u, v \in V$, there exists a path between u and v.

For any two nodes $u, v \in V$, $d(u, v)$ denotes the Euclidean distance between u, v. For any node u, $N(u)$ denotes the set of neighbors of u. That is, $N(u) = \{v \in V | (u, v) \in E\}$. A set $T \subseteq V$ is said to be a cover for a set $R \subseteq V$, if $R \subseteq \cup_{w \in T} N(w)$.

An independent set I in G is a subset of V such that $\forall u, v \in I, (u, v) \notin E$. A Maximal Independent Set U is an independent set which is not a proper subset of any other independent set. Any MIS U is clearly a cover for $V \setminus U$, since any uncovered node $v \in V \setminus U$ could be added to U, which would contradict its maximality.

Given a source node $s \in V$, the depth of any node in G with respect to s is the minimum number of hops between s and v. The radius of G with respect to s, denoted by R, is the maximum depth of all nodes in G with respect to s.

The minimum degree of G is denoted by $\delta(G)$. The inductivity of G is defined by $\delta^*(G) = max_{U \subseteq V} \delta(G[U])$. A proper node coloring of G is an assignment of colors, represented by natural numbers, to the nodes in V such that any pair of adjacent nodes receive different colors. It is well known that one is able to produce a proper node coloring in G using at most $1 + \delta^*(G)$ colors. This is accomplished by using a smallest-degree-last ordering of nodes and assigning proper colors to nodes in that particular order [11].

4.2 Network Model

We assume that all the nodes in the network have an equal transmission range, an equal interference range, and an equal carrier sensing range. Therefore, the network is represented by a UDG $G = (V, E)$. For all nodes it is assumed that the transmission range is equal to r, the interference range is equal to αr, $\alpha \geq 1$, and the carrier sensing range is equal to βr, $\beta \geq 1$. Two nodes u, v are connected in G, iff $d(u, v) \leq r$.

Two transmissions "$t_1 \rightarrow r_1$" and "$t_2 \rightarrow r_2$" are said to be parallel if they are scheduled in the same time slot. In order for such parallel transmissions to be non-conflicting, they must avoid all three types of conflict discussed in Section 2.1. More specifically, all the following conditions must be satisfied:

Condition 1: $d(t_1, r_2) > \alpha r$

Condition 2: $d(t_2, r_1) > \alpha r$

Condition 3: $d(t_1, t_2) > \beta r$

The first two conditions avoid Type 1 and Type 2 conflicts between transmitters and receivers of different transmissions. The third condition avoids Type 3 conflicts.

4.3 Problem Statement

We are given a UDG $G = (V, E)$, a source node $s \in V$, and parameters r, α, β, where $\alpha \geq 1, \beta \geq 1$. Node s is assumed to have the message before the broadcast is requested. The problem is to compute a minimum-latency conflict-aware broadcast schedule from source node s to all other nodes in the network. The schedule must avoid all three types of conflict outlined in the network model.

We assume that message transmissions proceed in synchronous time slots. Therefore, the schedule can be represented as an assignment of node transmissions to time slots. More specifically, the schedule assigns to each node $u \in V$ a time slot $t(u)$ at which u broadcasts the message. If a node does not transmit the message, no time slot is assigned to it. In order to have a valid schedule, all nodes in the network except for s must be informed by some node. Also a node u can only transmit the message at time $t(u)$ only if it has already been informed at time t' such that $t' < t(u)$ and $\nexists v$ such that $t(u) = t(v)$ and parallel transmissions by v and u are conflicting. The objective is to minimize $max(t(u)), u \in V$.

5. APPROXIMATION ALGORITHM

In this section we present the approximation algorithm called **Conflict-Aware Broadcast Scheduler (CABS)**. Before presenting the algorithm, in the following subsection we discuss the mechanism it employs for avoiding conflicting transmissions. Discussion of the algorithm and its correctness and ratio analysis are presented in subsequent subsections.

5.1 Mechanism for Avoiding Conflict

CABS's mechanism for creating non-conflicting schedules is based on using two *Conflict Graphs*. These graphs are constructed in the beginning of the algorithm and subsequently consulted when CABS needs to schedule nodes for transmission. These two conflict graphs are denoted by G_{C_t} and G_{C_r}. They are constructed as follows: The graphs have the same vertices as the original graph G. Each conflict graph is associated with a *conflict radius*, which determines the edges in that graph. In G_{C_t} there is an edge between two nodes u, v iff $d(u, v) \leq max(\alpha + 1, \beta)r$. In G_{C_t} there is an edge between two nodes u, v iff $d(u, v) \leq (max(\alpha, \beta) + 2)r$. The following lemma shows the rationale behind creating these two graphs.

LEMMA 1. *In order for two parallel transmission "$t_1 \rightarrow r_1$" and "$t_2 \rightarrow r_2$" to be non-conflicting according to the network model given in Section 4.2, it is **sufficient** to have:* $d(t_1, t_2) > max(\alpha + 1, \beta)r \vee d(r_1, r_2) > (max(\alpha, \beta) + 2)r$.

PROOF. We know that r_1, r_2 must be within the transmission ranges of t_1, t_2, respectively. Therefore, we have $d(t_1, r_1) \leq r$, $d(t_2, r_2) \leq r$.

If we have $d(t_1, t_2) > max(\alpha+1, \beta)r$, as shown in Figure 2, using a triangle inequality we obtain $d(t_1, r_2) > max(\alpha + 1, \beta)r - r \geq \alpha r$, and likewise, $d(t_2, r_1) > \alpha r$. These two inequalities satisfy Condition 1 and Condition 2 in the network model. Condition 3 also directly follows from the assumption: $d(t_1, t_2) > max(\alpha + 1, \beta)r \geq \beta r$.

If we have $d(r_1, r_2) > (max(\alpha, \beta)+2)r$, as shown in Figure 3, using a triangle inequality we obtain $d(t_1, r_2) > (max(\alpha, \beta) + 2)r - r \geq (\alpha + 1)r > \alpha r$, and likewise, $d(t_2, r_1) > \alpha r$. These two inequalities satisfy Condition 1 and Condition 2 in the network model. From the same inequalities, we can obtain $d(t_1, r_2) > (\beta + 1)r$. Now, using a triangle inequality in the triangle between t_1, t_2, r_1 we get: $d(t_1, t_2) > (\beta+1)r-r = \beta r$, which establishes Condition 3. □

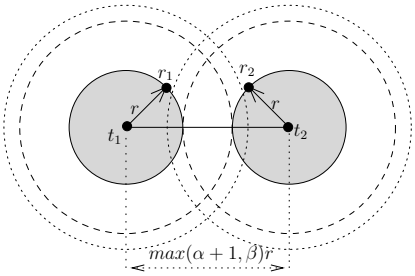

Figure 2: Conflict avoidance based on transmitters

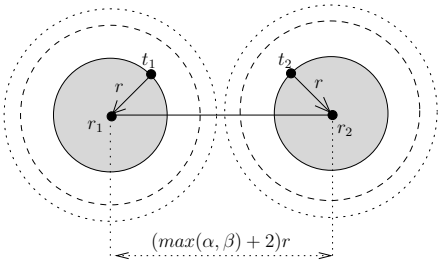

Figure 3: Conflict avoidance based on receivers

Note that these conditions are in general stronger than what is needed for avoiding conflicting transmissions. That is, it is possible for two transmissions not to have any conflicts even if they do not satisfy any of these two conditions.

The following corollary follows immediately from Lemma 1 and the construction of conflict graphs G_{C_t} and G_{C_r}:

COROLLARY 1. *In order for two parallel transmission "$t_1 \rightarrow r_1$" and "$t_2 \rightarrow r_2$" to be non-conflicting according to the network model given in Section 4.2, it is **sufficient** to have:* $(t_1, t_2) \notin E(G_{C_t}) \vee (r_1, r_2) \notin E(G_{C_r})$.

5.2 Algorithm Description

CABS uses a layer-by-layer technique to compute the broadcast schedule. The process starts with creating a Breadth First Search (BFS) tree rooted at source node s. All the nodes in the network are then partitioned into a set of layers according to their depths in the BFS tree. CABS informs nodes at some depth i only after it has informed all the nodes at depths 0 to $i - 1$. A Maximal Independent Set is formed along the BFS tree, and the broadcast progresses along the nodes in this independent set. For each layer i, first a set of transmissions are used to inform the nodes from the independent set at that layer. Then, those independent nodes are scheduled to inform their neighbors. All parallel transmissions are scheduled by consulting the conflict graphs G_{C_t} and G_{C_r} to ensure that they are non-conflicting. The following paragraphs explain the algorithm in more detail.

Once the BFS tree T_{BFS} has been formed, the nodes are partitioned into layers $L_0, L_1, ..., L_R$ according to their depths in the BFS tree. R denotes the radius of graph G with respect to s, which is equivalent to the height of the BFS tree. Then, an MIS U is formed induced by a non-decreasing order of depth of nodes in T_{BFS}. That is, the nodes are considered for inclusion in U based

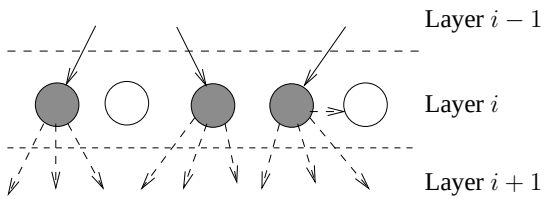

Figure 4: Scheduling broadcast in layer i

on a non-decreasing order of depth in the BFS tree. Let U_i denote the set of all nodes in U which are at layer i in T_{BFS}. As a result of this partitioning, all the nodes in U are partitioned into $R+1$ disjoint subsets $U_0, U_1, ..., U_R$. The algorithm then uses $R+1$ iterations to broadcast the message layer by layer using the nodes in $U_0, ..., U_R$, such that after each iteration i, all nodes at layer i are informed. During each iteration i, two tasks are performed. First, a cover UC_i of U_i is formed, such that all the nodes in UC_i are at layers up to $i-1$, and thus have been informed before the start of iteration i. Then, the auxiliary procedure **Sub-CABS** is used to produce a sub-schedule for delivering the message from UC_i to U_i. The second step in each iteration is producing a sub-schedule for broadcasting the message from U_i to all their uninformed neighbors in G. This task is handled by Sub-CABS as well. Figure 4 depicts the two steps involved in each iteration. Gray circles represent nodes in U_i and white circles represent other nodes at layer i in T_{BFS}. Solid arrows represent transmissions scheduled for informing nodes in U_i. Dashed arrows represent the transmissions scheduled in the second step, where nodes in U_i inform all their uninformed neighbors. Algorithm 1 shows the pseudocode for CABS.

Algorithm 1 Conflict-Aware Broadcast Scheduler

Procedure **CABS**$(G=(V,E), r, \alpha, \beta, s)$

1: $G_{C_t} \leftarrow$ **Conflict-Graph**$(G, max(\alpha+1, \beta)r)$
2: $G_{C_r} \leftarrow$ **Conflict-Graph**$(G, (max(\alpha, \beta)+2)r)$
3: $T_{BFS} \leftarrow Breath\text{-}First\text{-}Search(G, s)$
4: $U \leftarrow MIS(G)$ induced by non-decreasing depths in T_{BFS}
5: $R \leftarrow Height(T_{BFS})$
6: Partition U into $U_0, ..., U_R$
7: **for** $i \leftarrow 0$ to R **do**
8: $UC_i \leftarrow \emptyset$
9: **for** $u \in U_i$ **do**
10: If $UC_i \cap N(u) = \emptyset$ then $UC_i \leftarrow UC_i \cup$ {any informed neighbor of u}
11: **end for**
12: **Sub-CABS**$(U_i, \leftarrow, UC_i, G_{C_r})$
13: $W_i \leftarrow$ set of all uninformed neighbors of U_i
14: **Sub-CABS**$(U_i, \rightarrow, W_i, G_{C_t})$
15: **end for**

Sub-CABS is a generic procedure which can produce a conflict-aware sub-schedule for delivering the message from a set of transmitting nodes to a set of receiving nodes, provided that the former is a cover for the latter. More specifically, it is given four parameters. The first parameter P is a set of independent nodes in original graph G. The third parameter Q specifies a second set of nodes. The second parameter determines the direction in which the message should be delivered. A symbolic value of "$\leftarrow$" for third parameter indicates that the message should be delivered from the nodes in Q to the nodes in P. Likewise, a symbolic value of "$\rightarrow$" indicates that the message should be delivered from the nodes in P to the nodes in Q. The last parameter passed to this procedure is a

Procedure 2 Broadcast Sub-Scheduler

Procedure **Sub-CABS**$(P, direction, Q, G_C)$

1: Find a proper node coloring of $G_C[P]$
2: $l \leftarrow$ number of colors used in 1.
3: $S_1, ..., S_l \leftarrow \emptyset$
4: **for** $u \in P$ **do**
5: **if** $direction =$"$\leftarrow$" **then**
6: $v \leftarrow$ any neighbor of u in Q
7: $S_{Color(u)} \leftarrow S_{Color(u)} \cup \{v\}$
8: **else if** $direction =$"$\rightarrow$" **then**
9: $S_{Color(u)} \leftarrow S_{Color(u)} \cup \{u\}$
10: **end if**
11: **end for**
12: Output non-empty sets $S_1, ..., S_l$

Procedure 3 Conflict Graph Constructor

Procedure **Conflict-Graph**$(G=(V,E), r_c)$

1: Create new graph $G'=(V', E')$
2: $V' \leftarrow V$, $E' \leftarrow \emptyset$
3: $\forall u, v \in V$, if $d(u,v) \leq r_c$ then $E' \leftarrow E' \cup (u,v)$
4: Return $G'=(V', E')$

conflict graph G_C. Depending on whether the independent nodes P are transmitting or receiving, G_C takes the value of G_{C_t} or G_{C_r}, respectively.

When Sub-CABS is requested to create a sub-schedule from the set UC_i to the set U_i, it is given G_{C_r} as the conflict graph. Sub-CABS creates the sub-schedule by creating a proper node coloring of $G_{C_r}[P] = G_{C_r}[U_i]$. Each node's assigned color index determines when it is scheduled to receive the message: All nodes with the same color are scheduled to receive at the same time slot. Nodes with different colors are scheduled at different time slots.

For the second step in each iteration, Sub-CABS is requested to create a sub-schedule from the set U_i to their neighbors. In this case, it is given G_{C_t} as the conflict graph. Sub-CABS creates the sub-schedule by creating a proper node coloring of $G_{C_t}[P] = G_{C_t}[U_i]$. Each node's assigned color index determines when it is scheduled to transmit: All nodes with the same color are scheduled to transmit at the same time slot.

After $R+1$ iterations, all the nodes in the network are going to successfully receive the message. Moreover, the created schedule will have no conflicting parallel transmissions. We prove these claims in the following section.

5.3 Correctness Analysis

In this section we prove the correctness of CABS. We also prove that the generated schedules are valid based on our network model.

In order to inform nodes in U_i in each iteration i, CABS uses a cover UC_i for U_i such that nodes in UC_i have already been informed before the start of iteration i. The following lemma states that such a cover can always be formed.

LEMMA 2. *In each iteration i, CABS is able to find a cover UC_i for U_i such that the nodes in UC_i are informed before iteration i. In other words, for any node u in U_i, there is at least one node $v \in V$ such that $(u,v) \in E$ and v has already been informed before the start of iteration i.*

PROOF. We prove the lemma by induction on i. The base case is trivially true for $i=0$ because $U_0 = \{s\}$, and s already holds the message before the algorithm starts. We assume that the lemma holds for layers 0 through $i-1$ and prove the statement for layer

i. Let u be any node in U_i. Consider any neighbor v of u in layer $i-1$. u is guaranteed to have at least one such neighbor. Clearly, $v \notin U_{i-1}$ because otherwise its neighbor, u, could have not been included in U_i. On the other hand, the specific construction of U induced by a non-decreasing order of depths of nodes in T_{BFS} dictates that v must have had some neighbor in $\cup_{0 \leq j \leq i-1} U_j$ which has prevented the inclusion of v in U_{i-1}. This implies that v is going to be informed by that neighbor by the second call to Sub-CABS in some previous iteration j, $0 \leq j \leq i-1$. Therefore, node v can be scheduled to inform u in iteration i. □

We now prove the correctness of Sub-CABS.

LEMMA 3. *Sub-CABS is correct, and produces schedules with non-conflicting parallel transmissions.*

PROOF. Correctness of Sub-CABS follows easily from its requirement that the transmitting set must be a cover for the receiving set.

For proving its adherence to the network model, we consider two separate cases:

Case 1. Sub-CABS is asked to deliver the message from some set Q to an independent set P. In this case, the conflict graph G_{C_r} is used. The procedure performs the partitioning based on a proper node color of $G_{C_r}[P]$. Consider two nodes p_1, p_2 which have received the same color. Due to the proper node coloring done, p_1, p_2 must be non-adjacent in $G_{C_r}[P]$. Due to the construction of $G_{C_r}[P]$, this implies that $d(p_1, p_2) > (max(\alpha, \beta) + 2)r$. Therefore, according to Lemma 1, any parallel transmissions to p_1, p_2 are non-conflicting. Therefore, they can receive the message at the same time slot from any of their respective neighbors in Q.

Case 2. Sub-CABS is asked to deliver the message from an independent set P to some set Q. In this case, the conflict graph G_{C_t} is used. The procedure performs the partitioning based on a proper node coloring of $G_{C_t}[P]$. Consider two nodes p_1, p_2 which have received the same color. Due to the proper node coloring done, p_1, p_2 must be non-adjacent in $G_{C_t}[P]$. Due to the construction of $G_{C_t}[P]$, this implies that $d(p_1, p_2) > max(\alpha + 1, \beta)r$. Therefore, according to Lemma 1, parallel transmissions from p_1, p_2 are non-conflicting. Therefore, they can broadcast in parallel to inform their neighbors in Q. □

We are now ready to show the correctness of CABS.

THEOREM 1. *Schedules produced by CABS are both correct and non-conflicting.*

PROOF. In each iteration, the second call to Sub-CABS informs all the nodes covered by U_i. Since $U = \cup_{0 \leq i \leq R} U_i$, all nodes in $V \setminus U$ are going to be informed after all iterations. Likewise, the nodes in U are informed as a result of the first calls to Sub-CABS in each iteration. In addition, the order of calls ensures that all nodes in U_i receive the message before they are scheduled to inform their neighbors. Therefore, CABS is correct.

Since all time slot assignments in CABS are actually done by Sub-CABS, from Lemma 3 it follows that CABS produces non-conflicting schedules. □

5.4 Performance Ratio Analysis

In this section we prove the approximation ratio of CABS. We compare the performance of our algorithm against the trivial lower bound of R, which is the radius of graph G with respect to s. In order to simply our analysis, for the time being we assume α to be equal to β. Under this assumption the conflict radii of graphs G_{C_t} and G_{C_r} change to $(\alpha+1)r$ and $(\alpha+2)r$, respectively.

In order to prove an upper-bound for CABS, we calculate an upper-bound on the number of time slots consumed by each invocation of Sub-CABS. Sub-CABS creates the schedules based on proper node coloring of $G_{C_t}[P]$ and $G_{C_r}[P]$. We prove that such a proper node coloring needs at most a constant number of colors, independent of the number of nodes in P or Q.

As noted in Section 4.1, the number of colors needed for a proper node coloring of G is bounded by $\delta^*(G) + 1$. So the analysis is reduced to bounding the inductivity of $G_{C_t}[P]$ and $G_{C_r}[P]$. The following lemma establishes this bound.

LEMMA 4. *The inductivities of $G_{C_t}[P]$ and $G_{C_r}[P]$ have an upper-bound of $O(\alpha^2)$ in each invocation of Sub-CABS.*

PROOF. First, consider $G_{C_t}[P]$. Two nodes u, v must be non-adjacent in $G_{C_t}[P]$ if we have $d(u, v) > (\alpha+1)r$. This constraint limits the possible region in which the neighbors of any node v in $G_{C_t}[P]$ can reside. We use this constraint to upper-bound the minimum degree of $G_{C_t}[P]$, which, in turn, upper-bounds the inductivity of this subgraph, according to the definition of inductivity given in Section 4.1.

Consider the bottom-most node v in this subgraph. All neighbors of v must lie in the half-annulus centered at v with radii $\frac{r}{2}$ and $(\alpha+1)r$ as shown in Figure 5.

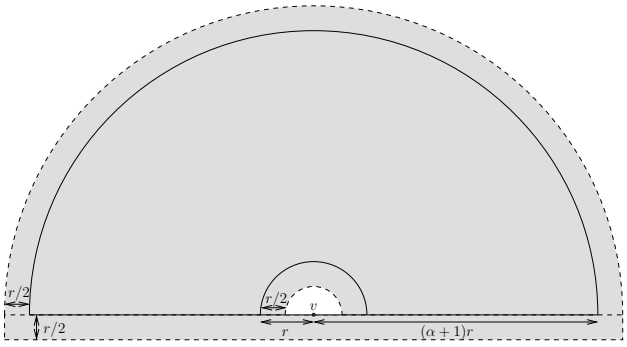

Figure 5: Bounding minimum degree of $G_{C_t}[P]$

The minimum degree of $G_{C_t}[P]$ cannot be more than the maximum number of nodes from P that can be a neighbor of v. Since the nodes in P are independent in G, their pairwise distances must be greater than r. For each node $p \in P$ consider a disk with radius $\frac{r}{2}$ centered at p. Let H denote the set of all such disks for all the neighbors of v in $G_{C_t}[P]$. Since the pairwise distances of the centers of these disks is greater than r, then the disks must be non-intersecting. Moreover, any point on a disk in H is at a distance at most $(\alpha+1)r + \frac{r}{2}$ from v. Therefore, the number of nodes in H is upper-bounded by the number of non-intersecting disks with radius $\frac{r}{2}$ whose areas are completely contained within the gray area shown in Figure 5. Therefore, the maximum value of $|H|$ is:

$$
\begin{aligned}
|H| &\leq \frac{\frac{1}{2}\pi\left((\alpha+\frac{3}{2})r\right)^2 + 2\left(\alpha+\frac{3}{2}\right)r\frac{r}{2} - \frac{1}{2}\pi\left(\frac{r}{2}\right)^2}{\pi\left(\frac{r}{2}\right)^2} \\
&= 2\left(\alpha+\frac{3}{2}\right)^2 + \frac{4}{\pi}\left(\alpha+\frac{3}{2}\right) - \pi \\
&= O(\alpha^2)
\end{aligned}
$$

Likewise, for $G_{C_r}[P]$, considering the area of a half-annulus with radii $\frac{r}{2}$ and $(\alpha+2)r$ gives a similar upper bound of $O(\alpha^2)$. □

As shown in the proof of this lemma, we rely on the independence of neighboring nodes for upper-bounding their number. This is why the node coloring is always done based on the independent set P.

The following corollary follows immediately from Lemma 4.

COROLLARY 2. *Each invocation of Sub-CABS uses at most $O(\alpha^2)$ time slots.*

We are now ready to prove the approximation ratio of CABS.

THEOREM 2. *The schedule generated by CABS is at most a constant ratio from the optimal solution.*

PROOF. CABS consists of $(R+1)$ iterations, each one of which makes two calls to Sub-CABS. Looking more carefully, we can observe that U_1 is always empty because $U_0 = \{s\}$ and none of s's neighbors at layer 1 can be in U due to s. Therefore, CABS needs exactly R iterations. On the other hand, according to Corollary 2, in each iteration Sub-CABS can produce a schedule within at most $O(\alpha^2)$ time slots. Therefore, the total number of time slots consumed by CABS is $O(\alpha^2)R$, which gives a constant approximation ratio. □

Assuming $\alpha \neq \beta$, the approximation ratio of the CABS turns out to be $O((max(\alpha, \beta))^2)R$.

5.5 Alternative Analysis

The argument in the previous section is based on bounding the area used by independent nodes in the half-annulus. We can provide a tighter analysis by partitioning the half-annulus into subregions such that each subregion can hold at most one independent node. Figure 6 shows one such possible partitioning.

The points $c_0, c_1, ..., c_n$ divide the line between c_0 and c_n into $n+1$ pieces. The distances of points $c_0, c_1, ..., c_n$ from v are $r_0, r_1, ..., r_n$ respectively, such that $\forall i : r_i \leq r_{i+1}$. Note that $r_0 = r$ and $r_n = (\alpha+1)r$. Using $n+1$ arcs centered at v and radii $r_0, r_1, ..., r_n$, we partition the original half-annulus into n smaller half-annuli. We denote the half-annulus between c_{i-1}, c_i by ρ_i. Each half-annulus ρ_i is further divided into k_i sectors as follows. We let $\beta_i = \frac{\pi}{k_i}$. We then draw $k_i + 1$ lines from v forming angles $k\beta_i$, $0 \leq k \leq k_i$ with line c_0c_n to create the sectors. All sectors created using these lines are going to be equal. Note that Figure 6 shows only one sector for each half-annulus. We denote the area of each sector in ρ_i by A_i.

In order for each sector to hold at most one independent node, the distance between any two points inside it must be at most r. In particular, length of the diagonal line connecting the farthest points on its perimeter must be at most r. One such diagonal is drawn for ρ_i in Figure 6. Using law of cosines, we should have:

$$r_{i-1}^2 + r_i^2 - 2r_{i-1}r_i \cos\beta_i \leq r, \quad 1 \leq i \leq n$$

In order to minimize k_i and get as few sectors as possible, we prefer this number to be as close to r as possible.

The area of each sector A_i is:

$$A_i = r_i^2\beta_i - r_{i-1}^2\beta_i, \quad 1 \leq i \leq n$$

The entire area of the original half-annulus, is:

$$A = \frac{1}{2}\pi((\alpha+1)r)^2 - \frac{1}{2}\pi r^2$$

Because the sum of the area of all sectors on all half-annuli must be equal to A, we obtain:

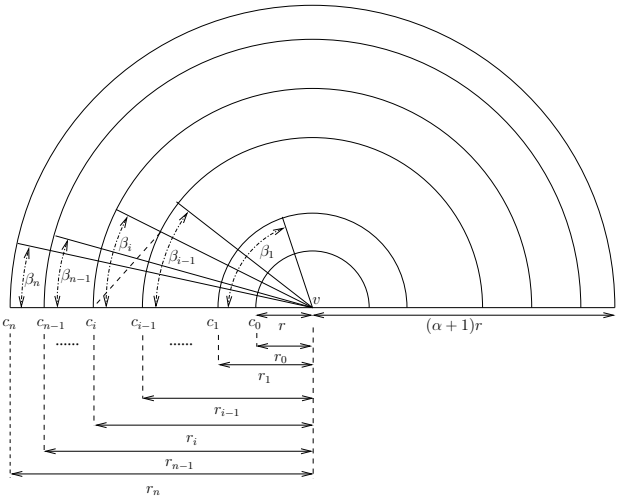

Figure 6: Partitioning neighborhood of v

$$A = \frac{\pi}{\beta_1}A_1 + \frac{\pi}{\beta_2}A_2 + ... + \frac{\pi}{\beta_n}A_n$$

Using the above constraints, together with a bound on n, like $n = 2(\alpha+1)$, we can create a programming system to determine optimal values for k_i, $1 \leq i \leq n$ with the objective of minimizing:

$$\sum_{i=1}^{n} k_i = \sum_{i=1}^{n} \frac{\pi}{\beta_i}$$

Note that we allow consecutive points c_{i-1}, c_i to coincide. In such case the half-annulus ρ_i vanishes, and we set $k_i = 0$.

The optimal value for the objective function in this system upper-bounds the number of independent nodes in the neighborhood of v. One way to solve this system is using the local maximum method and maximizing A_i for each i. This in turn determines values of β_i, r_i. Although solving this system gives a better constant compared to the method in Section 5.4, the constant is still $O(\alpha^2)$. This is to be expected because asymptotically the area A of the neighborhood grows as rapidly as α^2, while the coverage area of an individual neighbor is independent of α and stays constant.

6. HEURISTIC GREEDY ALGORITHM

In this section we present **HCABS**, which is an alternative algorithm for computing a broadcast schedule in G. Although we do not prove a bound on this algorithm, the experimental evaluation presented in Section 7 demonstrates its ability to generate efficient schedules.

HCABS tries to create more efficient broadcast schedules by optimizing its operation in three different areas. Firstly, instead of using the conflict graphs to determine conflicting parallel transmissions, it uses a manual conflict avoidance technique which is based on checking individual transmitters for violation of any of the necessary conditions in our network model. This helps increase the number of parallel transmissions which can be scheduled in each time slot.

Secondly, HCABS does not follow a layer-by-layer approach, where all the nodes in a BFS layer must be informed before the broadcast can proceed to the subsequent layers. Instead, HCABS considers the set of all informed nodes at any point in time as potential transmitters. By scheduling parallel transmissions in multiple layers, we can take advantage of the spatial distribution of the

transmitters to schedule more non-conflicting transmissions in each time slot.

The third optimization in HCABS is in deciding how to break the ties between conflicting transmissions at any step. Almost all approximation algorithms and heuristic algorithms discussed so far [1, 6, 9] use some criteria to give priority to particular transmissions in a set of conflicting transmissions. Usually, the priority is given to the transmitters with more neighbors in the network, or the transmitters with more children in the BFS tree. In order to create a more efficient algorithm, we designed a few candidate criteria to decrease the latency of the broadcast.

For instance, we made the observation that most of the time the source node is not located in the center of the network. In such cases, there is a region in the network which is the farthest from the source. The nodes in this region are most probably the last nodes that are informed during the broadcast. Therefore, they directly determine the latency of the broadcast. Consider the communication routes that connect the source to this region in the network. One can imagine that prioritizing the expansion of the broadcast along these routes can decrease the broadcast latency, as any delays along these routes can directly increase the final latency. Based on this observation, we experimented with giving priority to the nodes which have the most number of descendants in the BFS tree, as such nodes are more likely to be on the routes to the farthest points in the network. Based on the same observation, we also tried giving priority to the transmitters with larger heights in the BFS tree.

On the other hand, since the ultimate goal of the broadcast schedule is to inform all the nodes in the network, we tried following a greedy rule to locally optimizing the progress rate of the broadcast by informing as many nodes as possible with each new transmission. This greedy rule gives priority to transmitters which have the highest number of uninformed neighbors at that point in time. The pseudocode for HCABS, which employs this greedy optimization, is given in Algorithm 4.

HCABS maintains the set of all informed nodes in the set $Active$. This set is initialized to include only the source. Then, a number of iterations are followed until all the nodes in the network are informed. During each iteration, a priority queue PR is initialized with the set of all nodes in $Active$. Transmitters are extracted from the priority queue one by one and are considered to be scheduled. After scheduling a transmission, the algorithm performs the checks in lines 10-12 to remove from PR any transmitters which would be conflicting with the just scheduled transmission for the same time slot. This process continues until as many transmissions as possible are scheduled for the current time slot, at which point the algorithm proceeds to the next iteration.

The simulation results in the next section show the performance of HCABS under any of the above-mentioned tie-breaking criteria.

Algorithm 4 Greedy Broadcast Scheduler

Procedure **HCABS**$(G = (V, E), r, \alpha, \beta, s)$

1: $Inf \leftarrow \{s\}, Active \leftarrow \{s\}, Time \leftarrow 0$
2: Priority Queue PR: $key(u \in PR) = |N(u) \setminus Inf|$
3: **while** $Inf \neq V$ **do**
4: $PR \leftarrow Active, S \leftarrow \emptyset$
5: **while** $PR \neq \emptyset$ **do**
6: $u \leftarrow$ Extract-Min(PR)
7: $Active \leftarrow Active \setminus \{u\}$
8: **if** $N(u) \setminus Inf \neq \emptyset$ **then**
9: From PR remove all nodes v whose transmissions would conflict with the scheduled transmission by u, as follows:
10: $\forall v \in PR$: $\forall w \in N(v) \setminus Inf$, if $d(u, w) \leq \alpha r$ then $PR \leftarrow PR \setminus \{v\}$
11: $\forall v \in N(u) \setminus Inf$: $\forall w \in PR$, if $d(w, v) \leq \alpha r$ then $PR \leftarrow PR \setminus \{w\}$
12: $\forall v \in PR$, if $d(u, v) \leq \beta r$ then $PR \leftarrow PR \setminus \{v\}$
13: Schedule u as follows:
14: $S \leftarrow S \cup \{u\}$
15: **for** $w \in N(u) \setminus Inf$ **do**
16: $Inf \leftarrow Inf \cup \{w\}$
17: $Active \leftarrow Active \cup \{w\}$
18: **end for**
19: **end if**
20: **end while**
21: $Time \leftarrow Time + 1$
22: Schedule S in time slot $Time$
23: **end while**

7. EXPERIMENTAL EVALUATION

This section presents the experimental evaluation of our proposed algorithms through simulations. To make comparison of results easier, the network settings and parameters were chosen to be comparable with simulations performed by Chen et al. [1] and Gandhi et al. [6]. More specifically, we placed the nodes in a square of side 500 meters. The transmission ranges of nodes were set to 100 meters. The number of nodes were varied from 10 to 300 in increments of 10. The interference range parameter α, and the carrier sensing range parameter β were both set to 2. In all the experiments, the network nodes were placed randomly within the square. All the experiments were performed on connected graphs. Each experiment was run 100 times and the average values were used to plot the figures.

We simulated three algorithms. These included the two algorithms presented in this paper plus the recently introduced IAB algorithm by Chen et al. [1], which they claim to be the best existing algorithm [1] for this problem based on their simulations. In all experiments, the three algorithms were run against the same set of random graphs.

The IAB algorithm was discussed in Section 3. It schedules the broadcast layer by layer according to the depths of the nodes in the BFS tree. In each iteration, some nodes at layer i are used to inform all the nodes at layer $i + 1$. IAB gives priority to transmitters which are closer to the source s. More specifically, it sorts transmitters based on *Euclidean hops* from source, i.e. $\lfloor d(u, s)/r \rfloor$. If two nodes have the same value, the node that covers a node with smaller Euclidean hops from the source is given priority. Any ties are broken using the number of covered nodes. Since IAB does not consider Type 3 conflicts, for the purpose of this evaluation we slightly modified its implementation to let it produce conflict-aware schedules. More specifically, we augmented its conflict checking code with a single line of pseudocode similar to line 12 in Algorithm 4. We also had to fix a bug in IAB to help it avoid all sources of Type 2 conflicts. More specifically, IAB was missing a line of pseudocode similar to line 11 in Algorithm 4.

Figure 7 is a plot of the latency of the schedules produced by IAB, CABS, and HCABS. It shows the average latency of the algorithms over the 100 sample graphs for any number of nodes. As shown in the figure, HCABS consistently outperforms the other two algorithms. In sparser networks (less than 70 nodes), IAB is slightly outperforming CABS. However, as the number of nodes increases, CABS performs much better than IAB. As we go above 200 nodes, both the approximation algorithm and the heuristic algorithm seem to be converging on some fixed values. However, IAB is showing a seemingly linear growth rate.

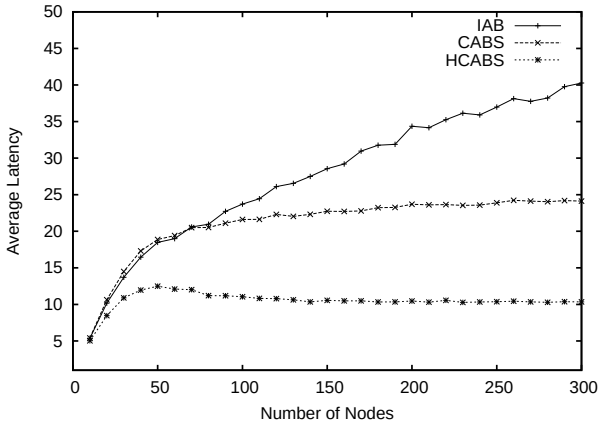

Figure 7: Average latency of algorithms

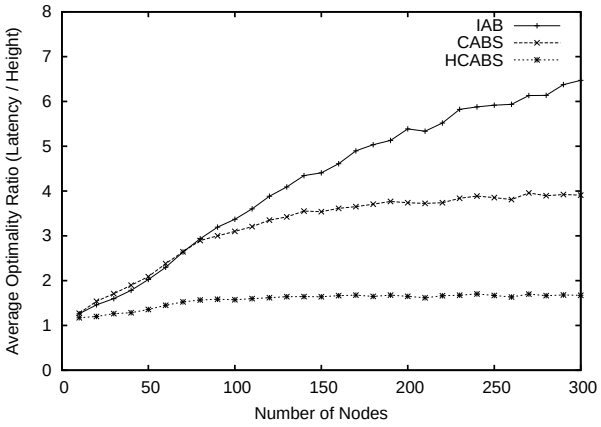

Figure 9: Average optimality ratio of algorithms

In fact, we expected all the algorithms to nearly converge to fixed values after a certain point. This is because intuitively the expected latency is mostly determined by the height of the BFS tree, rather than by the number of nodes. But, since the area of our network is limited, after a certain point the height of the BFS tree cannot grow any further. Since a single transmission can inform all the nodes in the neighborhood of the transmitter, an increase in the number of nodes without an impact on the height of the BFS tree should not add much to the latency. Figure 8 shows the average height of the BFS tree for any given number of nodes in our sample graphs. As seen in this figure, beyond about 130 nodes, the height of the BFS tree converges to about 6.25. The BFS height can be higher for some sparser networks, since the lower connectivity among nodes in sparser networks can result in some longer paths.

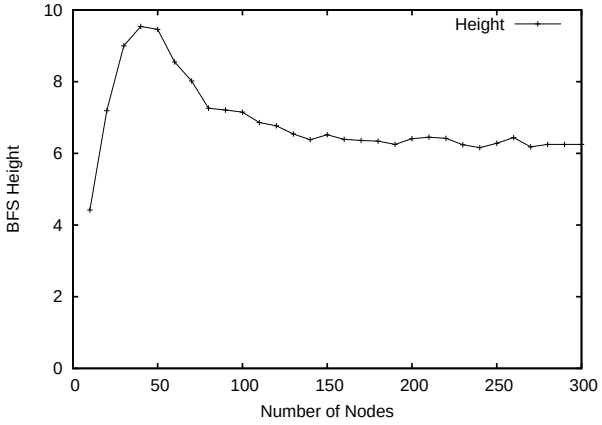

Figure 8: Average BFS height

Figure 9 plots the average *optimality ratio* of the broadcasts for the three algorithms. The optimality ratio is calculated by dividing the latency by the height of the BFS tree, which serves as a trivial lower bound for the algorithms. The average optimality ratios of IAB, CABS, and HCABS on all 3000 tested graphs were 4.25, 3.22, and 1.57, respectively. CABS's average performance is 32% better than IAB. HCABS's average performance is, in turn, 105% better than CABS. We remark that we could improve the efficiency of CABS by adapting the optimizations introduced in HCABS. However, we left this improvement out so that we can better contrast the operation of the algorithms.

There are two major reasons for the better performance of the heuristic algorithm in comparison with the approximation algorithm. Firstly, the approximation algorithm uses a layer-by-layer approach, whereas the heuristic algorithm schedules parallel transmissions in more than one layer. The greater distances between transmitters in different layers lowers the probability of encountering conflicts. Secondly, the approximation algorithm uses a conservative approach to avoiding conflicts, which is based on using the two conflict graphs G_{C_t}, G_{C_r}, as discussed in Section 5.1. This is while the heuristic algorithm employs a finer conflict avoidance technique, which can increase the number of parallel transmissions.

Comparing CABS with IAB, we can see how this conservative conflict avoidance technique makes CABS perform slightly worse than IAB on sparser graphs. IAB uses a manual conflict avoidance technique similar to HCABS. As the network becomes more dense, the manual conflict avoidance technique produces outputs which are similar to the conservative technique. This is because the probability of actually encountering a node in the areas which are blindly avoided by the conflict graphs approaches one. However, overall, CABS's performance is better than IAB, especially in denser networks. One factor in favor of CABS is that its broadcast proceeds along an independent set. Since the nodes in the independent set are relatively farther from each other, the probability that they encounter conflicts during either reception or transmission decreases. Moreover, in CABS, nodes in the independent set inform not only their children in the lower layer, but also some of their neighbors at the same layer. This makes the set of receivers for each iteration more scattered as well, which, in turn, increases the opportunity for scheduling non-conflicting transmissions for informing them. This is while for IAB all receivers are at the same layer and conflicts are more likely to occur in parallel transmissions.

However, we identified the main reason behind IAB's poor performance to be in its tie-breaking criteria, which prioritizes transmitters that closer to the source. Figure 10, which shows the state of the network at some point during the broadcast, helps explain this phenomenon. The shaded regions represent parts of the network which are informed. Only the nodes which reside in the outer shaded region have uninformed neighbors and are potential transmitters. Such nodes are represented by dark circles, while white circles represent the uninformed nodes. Among the potential transmitters, IAB prioritizes the ones which are relatively closer to the source. However, when a node is closer to the source, the uninformed area that it can inform becomes smaller. Such transmitters can inform as few as one or two nodes, even in a dense network.

Nonetheless, due to conflict avoidance, they prevent the more efficient transmitters in the outer parts from being scheduled in the same time slot. This effect slows down the expansion of the broadcast. In addition, since transmitters are always being chosen from the most interior parts of the informed region, a high percentage of all nodes in network are scheduled to transmit at some point. This is why IAB's performance seems to be proportional to the number of nodes.

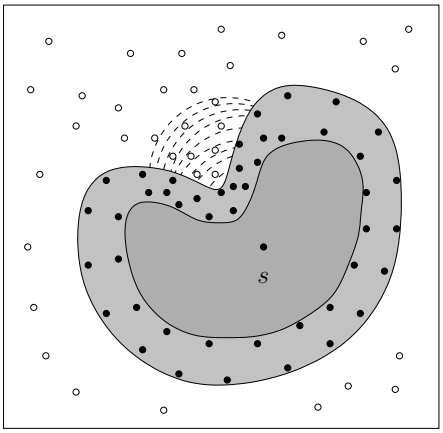

Figure 10: Demonstration of the slow expansion phenomenon

To gain better insight into this phenomenon, we devised an experiment. We designed a number of similar heuristic algorithms that work exactly like HCABS, but use different criteria to prioritize transmitters. In addition to the criteria mentioned in Section 6, we experimented with prioritizing transmitters based on the following measures: distance from source, Euclidean hops from source, number of neighbors, depth in the BFS tree, and finally, based on node IDs to simulate a random selection.

Table 1 compares the performance of these heuristic algorithms. For each instance, the average optimality ratio over all the 3000 sample graphs is listed. It is interesting to see that many of the chosen optimizations actually degrade the performance of the algorithm. This is because any criteria which is based on the topology of the network, is likely to put physically adjacent nodes of the network in close positions in the priority order as well. As shown in Figure 10, this can aggravate the slow expansion phenomenon. Table 1 also shows that it is better to prioritize nodes which are farther from the source rather than the nodes which are closer to the source. This is in line with the observation that farther nodes are more effective in rapidly expanding the informed region of the network during the broadcast.

8. CONCLUSIONS

The interference range and the carrier sensing range are two important parameters that can determine conflicts between parallel transmissions in wireless networks. In this paper we employed these parameters to develop a conflict-aware network model, which we used to study the problem of minimum-latency broadcast scheduling. We presented a constant approximation algorithm for this problem under our network model. The correctness of our algorithm and its approximation ratio were proven in detail. We also provided a more efficient heuristic algorithm for the same problem. Experimental results were provided to evaluate the performance of our solutions and show that they outperform existing algorithms. In addition, we analyzed the discussed algorithms to justify their performance trends.

Table 1: Comparison of heuristic algorithms

Selection Criteria	Avg. Ratio
Larger number of uninformed neighbors (HCABS)	1.57
Larger number of BFS descendants	2.26
Larger Euclidean hops to source	2.38
Larger BFS depth	2.41
Larger node ID (Random)	2.43
Larger BFS height	2.47
Larger number of neighbors	2.60
Larger distance to source	2.69
Larger number of neighbors	2.60
Smaller Euclidean hops to source (IAB)	3.28
Smaller BFS depth	3.35
Smaller distance to source	5.24

9. REFERENCES

[1] Z. Chen, C. Qiao, J. Xu, and T. Lee. A constant approximation algorithm for interference aware broadcast in wireless networks. In *INFOCOM*, pages 740–748, 2007.

[2] I. Chlamtac and S. Kutten. On broadcasting in radio networks - Problem analysis and protocol design. *IEEE Transactions on Communications*, 33:1240–1246, dec 1985.

[3] I. Chlamtac and S. Kutten. Tree-based broadcasting in multihop radio networks. *IEEE Transactions on Computers*, C–36(10):1209–1223, 1987.

[4] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1-3):165–177, December 1990.

[5] J. Deng, B. Liang, and P. Varshney. Tuning the carrier sensing range of ieee 802.11 mac. In *GLOBECOM*, pages 2987–2991, 2004.

[6] R. Gandhi, S. Parthasarathy, and A. Mishra. Minimizing broadcast latency and redundancy in ad hoc networks. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 222–232, 2003.

[7] L. Gasieniec, D. Peleg, and Q. Xin. Faster communication in known topology radio networks. In *PODC '05: Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, pages 129–137, 2005.

[8] C. Ho, K. Obraczka, G. Tsudik, and K. Viswanath. Flooding for reliable multicast in multi-hop ad hoc networks. In *Proceedings of the 3rd Int'l Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 64–71, 1999.

[9] S. C.-H. Huang, P.-J. Wan, X. Jia, H. Du, and W. Shang. Minimum-latency broadcast scheduling in wireless ad hoc networks. In *INFOCOM*, pages 733–739, 2007.

[10] J. Jercheva, Y. Hu, D. Maltz, and D. Johnson. A simple protocol for multicast and broadcast in mobile ad hoc networks, July 2002.

[11] D. W. Matula and L. L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *J. ACM*, 30(3):417–427, 1983.

[12] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *MobiCom '99*, pages 151–162. ACM Press, 1999.

[13] X. G. Viennot. A strahler bijection between dyck paths and planar trees. *Discrete Math.*, 246(1-3):317–329, 2002.