Multi-Channel Scheduling and Spanning Trees: Throughput-Delay Trade-off for Fast Data Collection in Sensor Networks

Amitabha Ghosh, Özlem Durmaz Incel, V. S. Anil Kumar, and Bhaskar Krishnamachari

***Abstract*—We investigate the trade-off between two mutually conflicting performance objectives – throughput and delay – for fast, periodic data collection in tree-based sensor networks arbitrarily deployed in 2-D. Two primary factors that affect the data collection rate (throughput) and timeliness (delay) are: (i) efficiency of the link scheduling protocol, and (ii) structure of the routing tree in terms of its node degrees and radius. In this paper, we utilize multiple frequency channels and design an efficient link scheduling protocol that gives a constant factor approximation on the optimal throughput in delivering aggregated data from all the nodes to the sink. To minimize the maximum delay subject to a given throughput bound, we also design an (α, β)-bicriteria approximation algorithm to construct a *Bounded-Degree Minimum-Radius Spanning Tree*, with the radius of the tree at most β times the minimum possible radius for a given degree bound Δ^*, and the degree of any node at most $\Delta^* + \alpha$, where α and β are positive constants. Lastly, we evaluate the efficiency of our algorithms on different types of spanning trees, and show that multi-channel scheduling, combined with optimal routing topologies, can achieve the best of both worlds in terms of maximizing the aggregated data collection rate and minimizing the maximum packet delay.**

***Index Terms*—Convergecast, TDMA scheduling, multiple channels, routing trees, approximation algorithms.**

I. Introduction

CONVERGECAST, namely the *many-to-one* flow of data from a set of sources to a common sink over a tree-based routing topology, is a fundamental communication primitive in sensor networks. Such data flows can be triggered either by external events, such as user queries to periodically get a snapshot view of the network, or can be automated over long durations. For real-time, mission-critical, and high data-rate applications [1]–[3], it is often critical to *simultaneously* maximize the data collection rate and minimize packet delays. In addition, when summarized information is required or the measurements are correlated, it is beneficial to aggregate data en route to the sink. This helps in reducing redundancy and the number of transmissions. We refer to such a data collection process under aggregation as *aggregated convergecast*.

Two primary factors that affect the data collection rate and packet delays are: (i) efficiency of the link scheduling protocol, and (ii) structure of the routing tree. A typical sensor node is equipped with a single half-duplex transceiver, using which it can either transmit or receive only one packet at any time. Moreover, nodes very close to each other cannot transmit simultaneously due to interference in the wireless medium. It is shown that for periodic traffic, multiple frequencies under spatial-reuse *time division multiple access* (TDMA) can eliminate interference and enable more concurrent transmissions [8], thus, enhancing the rate and providing bounds on the completion time of convergecast [12]. In addition, since TDMA protocols assign a dedicated time slot for each node to transmit and allow it to enter sleep modes during inactive periods, they perform well even under heavy traffic conditions and achieve low duty cycles. We note that, although multiple frequencies have been used in the domain of ad hoc networks, their use in sensor networks is new and challenging, especially due to resource constraints on the nodes. However, since current sensor network hardware, such as CC2420 radios, already support multiple frequencies, it is imperative that we take their advantage in designing provably-efficient, multi-channel TDMA scheduling protocols.

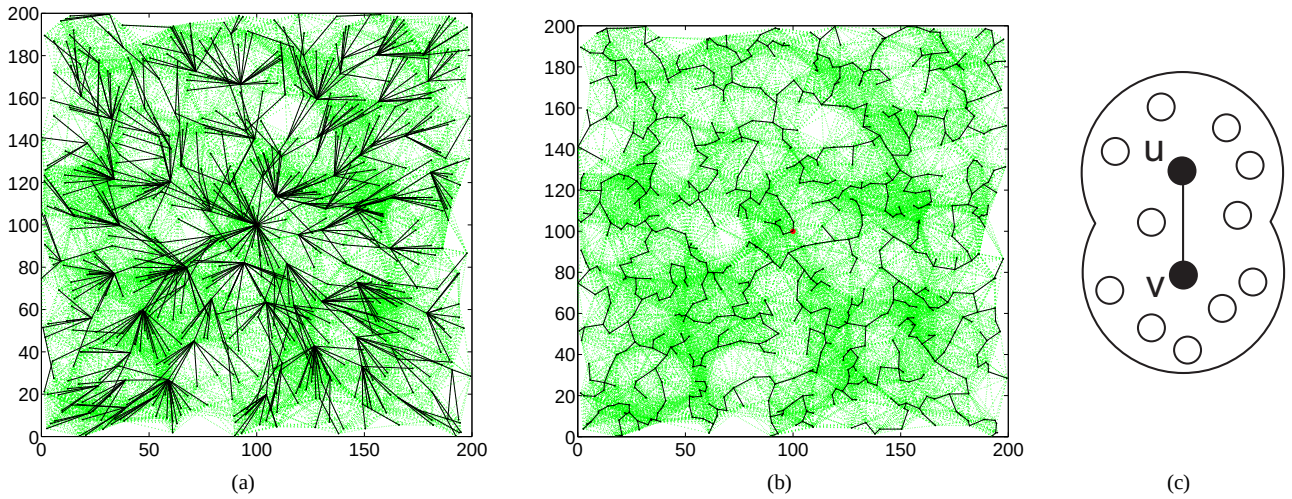
In [8], the authors show that once interference is reduced using multiple frequencies, the structure of the routing tree plays an important role in scheduling. It is shown that degree-constrained trees even with a single channel perform better than shortest-path trees (which have high degrees) with multiple channels. While it is true that the overall scheduling performance jointly depends on frequency-timeslot assignment and the routing tree structure, once multiple frequencies are used to eliminate interference, high node degree becomes the next major bottleneck in achieving high throughput, because the children of a common parent need to be scheduled at different time slots due to half-duplex radios. On the other hand, trees with low node degrees avoid bottlenecks and allow for more concurrent transmissions in the presence of multiple frequencies. For a given deployment of nodes, however, a spanning tree with low node degrees has large hop distances to the sink. Thus, if packet delays are measured purely in terms of hop counts, a tree with low node degrees is likely to incur high delays as opposed to one with high node degrees. These two opposing factors - *node degree* and *hop distance* - therefore, underscore the importance of the routing topology in maximizing the rate and minimizing packet delays.

Fig. 1(a) shows a *shortest-path tree* (SPT) on a network of 800 nodes randomly deployed in a region of size 200×200. The sink is located at the center, and a link between any two

A. Ghosh and B. Krishnamachari are with the Dept of Electrical Engineering, University of Southern California, LA, {amitabhg, bkrishna}@usc.edu

Ö. D. Incel is with NETLAB, Dept of Computer Engineering, Bogazici University, Turkey, ozlem.durmaz@tam.boun.edu.tr

V. S. Anil Kumar is with the Dept of Computer Science and Virginia Bio-Informatics Institute, Virginia Tech, Blacksburg, akumar@vbi.vt.edu



(a) (b) (c)

Fig. 1. (a) Shortest Path Tree (SPT): high node degrees but minimum hop distances to the sink. (b) Minimum Interference Tree (MIT): low node degrees but large hop distances to the sink. Dark lines represent tree edges, dotted lines represent interfering links on the same communication graph. (c) Cost of edge (u, v) is 11.

nodes exists if they are within a distance of 25 from each other. We observe that the nodes in the SPT have high degrees but minimum hop distances to the sink. Fig. 1(b) shows a *minimum spanning tree* (MST) on the same deployment, where the cost of an edge (u, v) is equal to the number of nodes covered by the union of the two disks centered at nodes u and v, each of radius equal to their Euclidean distance $d(u, v)$ (cf. Fig. 1(c)). This cost function gives a measure of the interference by counting the number of nodes affected by u and v communicating with just enough transmit power to exactly reach each other. The MST thus constructed is known as the *minimum interference tree* (MIT) [4], which clearly has low node degrees but large hop distances to the sink. Thus, if an SPT is best for achieving low delays, an MIT is more suitable for high data collection rate. However, we note that the designer of a scheduling algorithm might not always have the flexibility to construct the best possible routing tree; sometimes, network designers/planners have specific constraints due to socio-economic reasons (e.g., cost constraints), and can allow data flow only along specific paths in the network. In such cases, the routing tree is fixed and given a-priori, for example, a minimum-cost spanning tree with the cost function depending on edge lengths and link bit error rates (BER). In addition, there might be topological constraints that force data to follow specific paths. For instance, in structural health monitoring, one can deploy nodes only at specific locations due to geometric constraints, and accordingly can have access to only fixed and pre-specified routing paths.

In this paper, we consider aggregated convergecast on *arbitrarily* deployed networks in 2-D, and design algorithms with provably-good performance bounds for *link scheduling* and constructing *routing topologies* to simultaneously maximize the data collection rate and minimize packet delays. More specifically, our key contributions are twofold: (i) for a given routing tree, we design a multi-channel link scheduling protocol that gives a constant factor approximation on the optimal aggregated data collection rate, and (ii) we design a bicriteria constant factor approximation algorithm to construct a routing tree minimizing the maximum hop distance to the sink (i.e., minimizes the maximum delay) for a given maximum degree constraint. To the best of our knowledge, this is one of the first works to simultaneously consider both throughput and delay under the same framework in wireless sensor networks.

The rest of the paper is organized as follows. Section II describes related works. In Section III, we describe our models, assumptions, and problem formulation. Section IV focuses on designing a multi-channel link scheduling algorithm for aggregated convergecast, and Section V presents an algorithm for constructing a bounded-degree, minimum-radius routing tree. We present our numerical evaluations in Section VI, and finally draw some conclusions in Section VII.

II. Related Work

The scheduling problem with the objective to minimize the number of time slots required to complete convergecast (known as the *schedule length*) has been studied in [5]–[9] for aggregated data, and in [10]–[12] for raw data. Most of the existing algorithms aim to maximize the number of concurrent transmissions and enable spatial reuse by devising strategies to eliminate interference.

For aggregated convergecast, Annamalai *et al.* [5], investigate the use of orthogonal codes to eliminate interference where nodes are assigned time slots from the bottom of a convergecast tree to the top. Similarly, in [6], the problem is defined as a *Minimum Data Aggregation Time* problem with the goal to find a collision-free schedule that routes data from the subset of nodes to the sink in the minimum possible time. These studies, however, consider one-shot data collection rather than continuous and periodic convergecast over long durations like in our case. In addition, they do not consider the impact of routing trees and instead focus on the *causality constraint* by which a node is not eligible to be scheduled before it receives all the packets from its children.

In [7], Moscibroda theoretically shows that non-linear power control mechanisms (without discrete power levels) can sig-

nificantly improve the scheduling complexity and capacity of wireless networks. In his work, the aggregated data capacity as well as the notion of worst-case capacity, which concerns the question of how much information can each node transmit to the sink regardless of the networks topology, are investigated for typical *worst-case* structures, such as chains. However, it does not consider further generalizations for convergecast trees and the trade-off between throughput and delay.

In case of raw data convergecast, Gandham *et al.* [12] consider the scheduling problem using a single channel TDMA protocol. They describe an integer linear programming formulation and propose a distributed scheduling algorithm that requires at most $3N$ time slots for general networks, where N is the number of nodes. A similar study [10] is presented by Choi *et al.* in which an NP-completeness result is proved on minimizing the schedule length for a single frequency.

The use of multiple channels has been well researched in the domain of ad hoc networks. To improve network throughput, So *et al.* propose a MAC protocol that switches channels dynamically and avoids the hidden terminal problem using temporal synchronization [13]. A link-layer protocol called SSCH is proposed by Bahl *et al.* that increases the capacity of IEEE 802.11 networks by utilizing frequency diversity [14]. In the domain of sensor networks, however, there exist fewer works using multiple channels. The first multi-frequency MAC protocol MMSN is proposed by Zhou *et al.* where the goal is to increase the aggregated throughput [15].

Several optimization problems arising in the design of communication networks can be modeled as constructing optimal network topologies [21], in particular, spanning trees that satisfy certain constraints on node degrees, diameter, or total cost. The *Minimum Degree Spanning Tree* problem, where the goal is to construct a spanning tree such that its maximum node degree is minimized, is NP-hard on general graphs [20]. The best known algorithm proposed Furer and Raghavachari [22] computes a spanning tree with maximum node degree at most $\Delta^* + 1$, where Δ^* is the optimum node degree. In [23], Singh and Lau consider the *Minimum Bounded Degree Spanning Tree* problem where, given a degree bound on each vertex, they find a spanning tree of optimal cost with each degree exceeding its bound by at most one. The *Minimum Diameter Spanning Tree* problem is to construct a spanning tree such that the tree diameter, defined as the longest hop distance between any pair of nodes, is minimized. On Euclidean graphs, this problem is solved in polynomial time $\Theta(N^3)$, and the result extends to any complete graph whose edge weights satisfy a distance metric [24]. The most recent result on general graphs is proposed in [25] that runs in $O(mN + N^2 \log N)$ time, where m is the number of edges.

Most closely related to our work is the *Bounded-Degree Minimum-Diameter Spanning Tree* problem, where the goal is to minimize the tree diameter subject to a degree constraint. The first bicriteria approximation algorithm on general graphs is proposed by Ravi *et al.* [26], which runs in $O(mN \log N)$ time and finds a spanning tree of degree $O(\Delta^* \log N + \log^2 N)$ and diameter $O(D \log N)$, where Δ^* is the minimum maximum degree of any spanning tree of diameter at most D. The authors use the notion of *poise* of a tree, defined as the maximum degree of any node plus the diameter, and use *multi-commodity flow* results to prove the approximations. For complete graphs, an $O(\sqrt{\log_{\Delta^*} N})$-approximation algorithm is proposed by Konemann *et al.* [27] under the Euclidean metric. It uses a combination of filtering and divide and conquer techniques to find a spanning tree of maximum node degree Δ^* and diameter $O(\sqrt{\log_{\Delta^*} N} \cdot D)$.

Our work differs from the above in that we consider the routing tree construction problem on *random geometric graphs*, where the goal is to minimize the radius of a spanning tree subject to a predefined budget on the degree. In our previous studies [8], [9], we had investigated the impact of transmission power control and multiple frequency channels on the schedule length. In this work, we further extend those results by studying the impact of routing trees on both maximizing the aggregated sink throughput and minimizing the maximum delay.

III. Preliminaries

A. Model and Assumptions

We model the network as an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges representing communication links. We assume that the network is connected, and all the nodes have a uniform transmission range R whose value depends on a signal-to-noise-ratio (SNR) threshold. Thus, any two nodes u and v can communicate with each other if their Euclidean distance $d(u, v)$ is at most R. We denote the sink by s, and define the *radius* of a spanning tree T on G rooted at s as the maximum hop distance from any node to the sink s. Each node is equipped with a single half-duplex transceiver, using which it can either transmit or receive a single packet at any given time.

We consider the *protocol interference model* (a.k.a. disk graph model), in which concurrent transmissions on two edges interfere with each other if and only if: (i) the edges are adjacent, or (ii) both the transmissions are on the same frequency, and at least one of the receivers is within the interference range of the non-intended transmitter. These two types of interferences are known as *primary* and *secondary* conflicts, as illustrated in Fig. 2(a) and 2(b), respectively. The setting of the interference range is empirically determined and is typically 2 to 3 times the transmission range [29]. In this work, we assume that it is η times R.

Under a TDMA setting, consecutive time slots are grouped into equal size frames that are repeated for periodic scheduling. We assume that every node generates a single packet in the beginning of each frame, and it has the ability to aggregate all the packets from its children as well as its own into a single packet before transmitting to its parent. The class of aggregation functions in this category include *distributive* and *algebraic* functions [16], where the size of an aggregated packet is constant regardless of the size of the raw measurements. Typical examples of such aggregation functions are MIN, MAX, MEDIAN, COUNT, SUM, AVERAGE, etc.

We assume that transmissions on different frequencies are orthogonal and non-interfering with each other. Although this assumption may sometimes fail in practice depending on

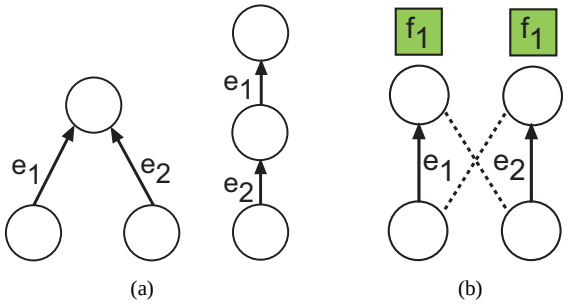


Fig. 2. (a) Primary conflict on adjacent edges e_1 and e_2. (b) Secondary conflict when transmissions are on the same frequency, say f_1, and at least one of the receivers is within the interference range of the non-intended transmitter.

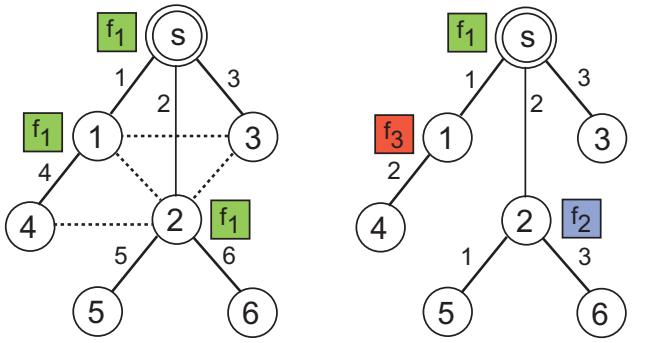
transceiver-specific adjacent channel rejection values, experimental results [8] show that scheduling performance remains similar for both CC2420 and Nordic nrf905 radios.

We consider a *receiver-based frequency assignment* strategy, in which we statically assign a frequency to each of the receivers (parents) in the tree, and have the children transmit on the same frequency assigned to their parent. Due to this static assignment, each node operates on at most two frequencies, thus, incurring less overhead compared to other dynamic assignments, such as pair-wise, per-packet negotiation of frequencies. This receiver-based channel assignment is a widely used approach in sensor networks as a convenient way to organize multi-channel protocols, because it simplifies synchronization issues as all receptions take place on the same channel at each node. For instance, using such a receiver-based strategy, a real-world implementation of a TDMA/FDMA solution for bulk data collection in sensor networks is presented in [32], while the maximum achievable rate for aggregated data collection is studied in [8]. In a recent work [35], the performance of receiver-based channel assignment is also compared with two other strategies called *tree-based multi-channel protocol* (TMCP) [30] and *joint frequency-timeslot scheduling* (JFTSS) [31], and is found to be superior than both. While JFTSS does not easily lend itself to a distributed solution (since interference relationships between all links must be known), in TMCP, contention inside the branches is not resolved because all the nodes on the same branch communicate on the same channel.

B. Problem Formulation

We first explain the process of aggregated convergecast and the notion of *schedule length*. Fig. 3(a) shows a network of 6 source nodes and a given routing tree whose edges are marked by solid lines; dotted lines represent secondary conflicts. We also show a possible frequency and time slot assignment.

The left-most column in Fig. 3(c) lists the receiver nodes (s, 1, and 2), and the entries in each row list the nodes from which packets are received by their corresponding receivers in each time slot. We note that at the end of frame 1, the sink has not yet received packets from nodes 4, 5 and 6, however, as the same schedule is repeated, aggregated packets from nodes 1 and 4, and nodes 2, 5, and 6 reach the sink starting from slot 1 and slot 2, respectively, of frame 2. The



	Frame 1						Frame 2					
	1	2	3	4	5	6	1	2	3	4	5	6
s	1	2	3	-	-	-	(1,4)	(2,5,6)	3	-	-	-
1	-	-	-	4	-	-	-	-	-	4	-	-
2	-	-	-	-	5	6	-	-	-	-	5	6

(c)

	Frame 1			Frame 2		
	1	2	3	1	2	3
s	1	(2,5)	3	(1,4)	(2,5,6)	3
1	-	4	-	-	4	-
2	5	-	6	5	-	6

(d)

Fig. 3. Aggregated convergecast: (a) Schedule length of 6 time slots with one frequency. (b) Schedule length of 3 time slots with two frequencies. (c), (d): Nodes from which aggregated data is received by their corresponding parents in each time slot over 2 consecutive frames for (a) and (b), respectively.

entries $(1, 4)$ and $(2, 5, 6)$ represent single packets comprising aggregated data. Thus, starting at frame 2, the sink continues to receive aggregated data from *all* the nodes once in every 6 time slots, and a *pipeline* is established. We measure the data collection *rate* by the number of time slots required to schedule all the tree edges exactly once per frame, and call it the *schedule length*. Maximizing the data collection rate is thus equivalent to minimizing the schedule length. In Fig. 3(b), we show the benefits of multiple frequencies by assigning different frequencies to the receiver nodes s, 1, and 2. This eliminates all secondary conflicts and reduces the schedule length to only 3 time slots, as shown in Fig. 3(d). We note that multiple frequencies cannot eliminate primary conflicts due to the inherent property of the transceivers being half-duplex.

Multi-Channel Scheduling Problem: Given a spanning tree T on G, and K orthogonal frequencies, we want to assign a frequency to each of the receivers, and a time slot to each of the edges in T such that the schedule length is minimized.

Since both node degree and tree radius affect the schedule length and packet delay, we formulate the problem of constructing routing trees as a *bicriteria* optimization problem [28], in which, given an upper bound on the maximum node degree, our goal is to minimize the tree radius. We call such a tree a *Bounded-Degree-Minimum-Radius Spanning Tree* (BDMRST). The routing tree construction problem is formally defined as follows.

Routing Tree Construction Problem: Given a graph G and

a constant parameter $\Delta^* \geq 2$, we want to construct a *Bounded-Degree-Minimum-Radius Spanning Tree* T on G rooted at sink s, such that the radius of T is minimized while the degree of any node in T is at most Δ^*.

We define an (α, β)-bicriteria approximation of the routing tree problem as one in which the maximum node degree is at most $\Delta^* + \alpha$, and the radius is at most β times the minimum possible radius subject to the degree constraint, where α and β are positive constants. We note that in our formulation, α is an *additive* factor whereas β is a *multiplicative* factor. Such bicriteria formulations are quite generic and robust, as the quality of approximation is independent of which of the two criteria the budget is imposed on, and it subsumes the case where one wishes to optimize a functional combination of the two objectives, such as, maximizing the sum or product of the maximum node degree and tree radius. Bicriteria optimization problems on spanning trees are often NP-hard on general graphs, and sometimes even on geometric graphs [28].

IV. Multi-Channel Scheduling

We observed in Fig. 3(b), that multiple frequencies, when assigned appropriately, can reduce the schedule length by eliminating secondary conflicts. In this section, our goal is to design a multi-channel link scheduling protocol that has a provably-good performance guarantee on the optimal schedule length. Formally, we define the decision version of the Multi-Channel Scheduling Problem on arbitrary graphs (where links can exist between any pair of nodes) as follows.

Multi-Channel Scheduling Problem (*decision version*): Given a routing tree T on an arbitrary graph G, and two positive integers p and q, is there an assignment of time slots to the edges of T using at most q frequencies to the receivers such that the schedule length is no more than p?

Theorem 1: Multi-Channel Scheduling Problem is NP-complete.

The proof follows from Theorem 3 in [9].

A. Scheduling With Unlimited Frequencies

The NP-hardness of the Multi-Channel Scheduling Problem is due to the presence of interfering links that cause secondary conflicts, making scheduling inherently difficult. This is because many subsets of non-conflicting nodes are candidates for transmission in each time slot, and the subset chosen in one slot affects the number of transmissions in the next slot. A natural question to ask, therefore, is to find the *minimum* number of frequencies that can eliminate *all* the secondary conflicts, which will then reduce the problem from being on a graph to being on a tree. This minimum, however, is again NP-hard to compute for arbitrary graphs, as shown in [17]. In the following, we give an upper bound, and show that when a *sufficient* number (i.e., unlimited) of frequencies is available, the scheduling problem can be solved optimally in polynomial time.

Create a *constraint graph* $G_C = (V_C, E_C)$ from the original graph G as follows (cf. Fig. 4(a) and 4(b) for an illustration). For each receiver (parent) in G, create a node in G_C. Connect any two nodes in G_C if their corresponding receivers in G are incident on two edges that form secondary conflicts.

Algorithm 1 BFS-TimeSlot-Assignment

1. Input: $T = (V, E_T)$
2. **while** $E_T \neq \phi$ **do**
3. $e \leftarrow$ next edge from E_T in BFS order;
4. Assign the minimum time slot to e respecting adjacency constraints;
5. $E_T \leftarrow E_T \setminus \{e\}$;
6. **end while**

Lemma 1: The number $K_{\max}$ of frequencies that will be sufficient to remove all the secondary conflicts in the original graph G is at most $\Delta(G_C)+1$, where $\Delta(G_C)$ is the maximum node degree in G_C. We note that this upper bound $\Delta(G_C)+1$ is a result of greedy coloring by first ordering the vertices, and it may not be tight.

Proof: Since we create an edge between every two nodes in G_C whenever their corresponding receivers in G form a secondary conflict, assigning different frequencies to every such receiver-pair in G is equivalent to assigning different colors to the adjacent nodes in G_C. Thus, $K_{\max}$ is equal to the minimum number of colors needed to vertex color G_C, called its *chromatic number*, $\chi(G_C)$. Since $\chi(G) \leq \Delta(G)+1$, for any arbitrary graph G, the lemma follows. ■

As illustrated in Fig. 4(b), the frequencies assigned to the receivers in G_C are as follows: frequency f_1 to nodes 1 and 2; f_2 to nodes 3, 4, and 8; f_3 to nodes s, 5, and 6; and f_4 to node 7. This particular frequency assignment is according to the heuristic called *Largest Degree First*, in which we consider the nodes in G_C in non-increasing order of their degrees and assign the first available frequency such that no two adjacent nodes have the same frequency.

Once all the secondary conflicts are eliminated by an appropriate frequency assignment to the receivers, the following time slot assignment scheme, called BFS-TimeSlot-Assignment (running in $O(|E_T|^2)$ time), presented in Algorithm 1 minimizes the schedule length. In each iteration (lines 2-6) of the algorithm, an edge e is chosen in the Breadth-First-Search order (starting from any node), and is assigned the minimum time slot that is different from all its adjacent edges. We prove in Theorem 2 that such an assignment gives a minimum schedule length equal to the maximum degree $\Delta(T)$ of T. An illustration is shown in Fig. 4(c).

Theorem 2: After all secondary conflicts are removed, Algorithm BFS-TimeSlot-Assignment gives a minimum schedule length equal to $\Delta(T)$.

Proof: The lower bound $\Delta(T)$ follows trivially, because the edges incident on the vertex with the maximum degree require at least $\Delta(T)$ distinct colors. For a BFS traversal on a tree, since an edge can conflict with at most $\Delta(T) - 1$ edges that come before it in the traversal, algorithm BFS-TimeSlot-Assignment uses $\Delta(T)$ colors. ■

B. Scheduling with Limited Frequencies

We showed in the previous subsection that all the secondary conflicts can be removed when sufficient frequencies are

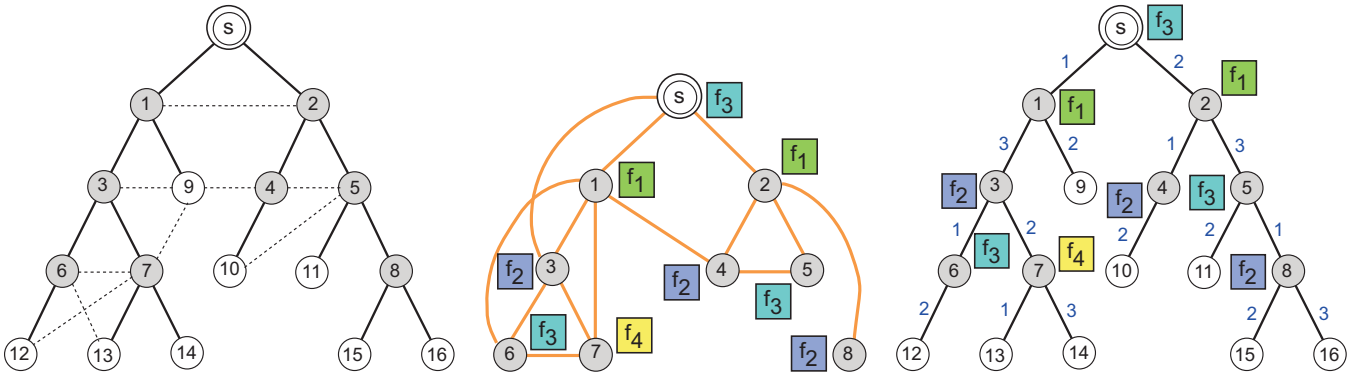


Fig. 4. (a) Original graph G; receiver nodes are shaded. (b) Constraint graph G_C and a frequency assignment to the receivers according to Largest Degree First. Here, 4 frequencies are sufficient to remove all the secondary conflicts, i.e., frequencies on adjacent nodes are different. (c) An optimal time slot assignment with schedule length 3 after all the secondary conflicts have been removed.

available, allowing us to compute a minimum-length schedule in polynomial time. However, typically there is a limitation on the number of frequencies over which a transceiver can operate, and, as shown in Theorem 1, the scheduling problem is NP-hard for a *given* (constant) number of frequencies. In this subsection, we take into account this constraint on the number of frequencies of current WSN hardware, and design an algorithm for the Multi-Channel Scheduling Problem that gives a constant factor approximation on the optimal schedule length.

We divide the 2-D deployment region into a set of square grid cells $\{c_i\}$, each of side length α. We define two cells to be *adjacent* to each other if they share a common edge or a common grid point. Thus, a cell can have either 3, 5, or 8 adjacent cells depending on whether it is a corner cell, an edge cell, or an interior cell, respectively. In our approach to design an algorithm for minimizing the schedule length, we decouple the frequency and time slot assignment phases. We first assign the frequencies to the receivers in T, such that the maximum number of nodes transmitting on the same frequency is minimized. Then, we employ a greedy time slot assignment scheme. We describe the two phases in detail below.

1) Frequency Assignment: Let $\mathcal{R}_i = \{v_1, \ldots, v_{n_i}\}$ denote the set of receivers on a given routing tree T that lie in cell c_i, and let $m : \mathcal{R}_i \rightarrow \{f_1, \ldots, f_K\}$ be a mapping that assigns a frequency to each of these receivers. Note that, $m(v_j) = f_k$ implies that all the children of v_j transmit on frequency f_k due to the receiver-based frequency assignment strategy.

DEFINITION 1: We define a *load-balanced frequency assignment* in cell c_i as an assignment of the K frequencies to the receivers in $\mathcal{R}_i$, such that the maximum number of nodes transmitting on the same frequency is minimized.

To express this formally, we define the *load* on frequency f_k in cell c_i under mapping m as the total number of children of all the receivers in $\mathcal{R}_i$ that are assigned frequency f_k, and denote it by $\ell_i^m(f_k)$. We call the number of children of node v_j its *in-degree*, and denote it by $deg^{in}(v_j)$. Thus,

$$\ell_i^m(f_k) = \sum_{v_j \in \mathcal{R}_i : m(v_j) = f_k} deg^{in}(v_j) \qquad (1)$$

Then, a load-balanced frequency assignment m^* in c_i is defined as:

$$m^* = \arg\min_m \max_k \{\ell_i^m(f_k)\} \qquad (2)$$

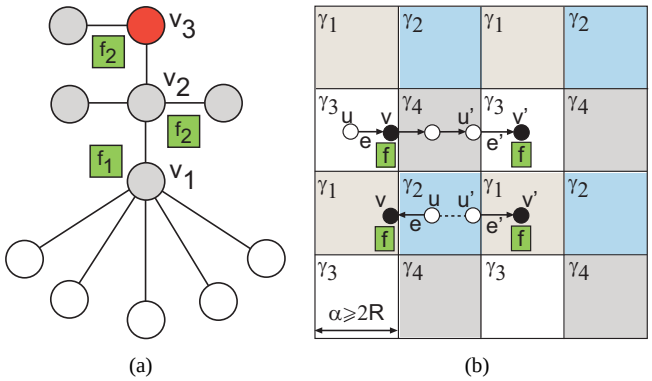
We denote the load on the maximally loaded frequency under mapping m^* in cell c_i by $\ell_i^{m^*}$. In the following lemma, we sketch a proof that load-balanced frequency assignment is NP-complete by reducing it from the well known problem of *Minimum Makespan Scheduling* (MMS) on identical parallel machines [19].

LEMMA 2: Load-Balanced Frequency Assignment (LBFA) is NP-complete.

Proof: Clearly, LBFA is in NP. Consider the following instance of the MMS problem. Given processing times of n jobs, $t_1, \ldots, t_n$, and an integer m, find an assignment of the jobs to m identical machines so that the completion time (*makespan*) is minimized. It is known that MMS is *strongly* NP-complete [19], and also admits a PTAS, originally due to Hochbaum and Shmoys [33].

Now consider the following reduction: For each job j, create one (receiver) node v_j, and place all of them within a single cell. Without loss of generality, assume that the times t_j's are integral. Then, for each node v_j, create t_j neighbors and place them in adjacent cells. Lastly, create one frequency for each machine. Clearly, the reduction runs in polynomial time (because MMS is strongly NP-complete). Then, for each cell, assigning the receivers to the frequencies in order to minimize the maximum load is the same as scheduling the jobs on the machines in order to minimize the makespan. It is easy to see that a solution for MMS corresponds to a solution of LBFA. Therefore, the lemma follows. ■

In Algorithm 2, we describe a frequency assignment scheme, called FREQUENCY-GREEDY, which gives a constant factor approximation on the optimal load. The basic idea of the algorithm is as follows: For each cell c_i, we sort the receivers in $\mathcal{R}_i$ in non-increasing order of their in-degrees; let this order be: $v_1, \ldots, v_{n_i}$. Then, starting from v_1, we assign to each subsequent node v_j a frequency that has the least load on it so far, breaking ties arbitrarily. In Fig. 5(a), we illustrate this scheme for two frequencies f_1 and f_2, and three receivers v_1, v_2, and v_3, sorted in non-increasing order of in-degrees. First,



(a) (b)

Fig. 5. (a) Frequency assignment according to FREQUENCY-GREEDY. Load on frequencies: $\ell(f_1) = 5$, $\ell(f_2) = 5$. White colored nodes transmit on frequency f_1; gray colored nodes transmit on frequency f_2. (b) Four pair-wise disjoint sets of time slots $\gamma_1, \gamma_2, \gamma_3$, and γ_4 schedule the whole network. Each set γ_i maps to a distinct color.

Algorithm 2 FREQUENCY-GREEDY

1. **for all** non-empty cell c_i **do**
2. Sort receivers in $\mathcal{R}_i$ in non-increasing order of in-degrees;
3. Suppose: $deg^{in}(v_1) \geq deg^{in}(v_2) \geq \ldots \geq deg^{in}(v_{n_i})$;
4. **for** $j = 1$ to n_i **do**
5. Find frequency f_k that is least loaded (breaking ties arbitrarily);
6. Assign f_k to v_j;
7. **end for**
8. **end for**

node v_1 is assigned frequency f_1, incurring a load of 5, as it has 5 children. Then node v_2 is assigned frequency f_2, giving a load of 3. Finally, node v_3 is also assigned frequency f_2, because f_2 is the least loaded so far. In this particular case, the assignment achieves an optimal load of 5 on both frequencies. In general, the following approximation holds.

LEMMA 3: Algorithm FREQUENCY-GREEDY in cell c_i gives a $\left(\frac{4}{3} - \frac{1}{3K}\right)$-approximation on the maximum load $\ell_i^{m^*}$ achieved by an optimal load-balanced assignment m^*.

Proof: We show that FREQUENCY-GREEDY is identical to Graham's list scheduling for MMS according to *longest processing time first* (LPT rule) [19]. Each receiver $v_j \in R_i$ corresponds to a job j, and its in-degree $deg^{in}(v_j)$ to time t_j (assume integral). Each frequency f_k corresponds to a processor m_k. The load on frequency f_k is therefore equal to the total time processor m_k takes. Since we first sort the receivers in non-increasing order of their in-degrees before assigning them to the least loaded frequency, it is identical to first sorting the jobs in non-increasing order of their processing times before assigning them to the least loaded machine. Since LPT gives a $\left(\frac{4}{3} - \frac{1}{3K}\right)$-approximation, therefore, so does FREQUENCY-GREEDY. ■

2) Time Slot Assignment: Once the receivers in each cell c_i are assigned frequencies according to algorithm FREQUENCY-GREEDY, we employ a greedy time slot assignment scheme for the whole network.

LEMMA 4: Let γ_i denote the *set* of time slots needed to schedule all the edges in cell c_i. Then, the minimum schedule length, Γ, for the whole network is bounded by: $\Gamma \leq 4 \cdot \max_i |\gamma_i|$, for any $\alpha \geq 2\eta R$.

Proof: Consider the grid cells shown in Fig. 5(b) for $\eta = 1$. Under the protocol interference model, a secondary conflict exists between any two nodes if they are within a distance ηR away from each other. This implies that interference is spatially restricted and time slots can be reused across cells that are well separated. In particular, for any $\alpha \geq 2\eta R$, two edges $e = (u, v)$ and $e' = (u', v')$, whose receivers v and v' are in non-adjacent cells, must have their non-intended transmitters u' and u, respectively, more than distance a ηR away and, therefore, can be scheduled on the same time slot regardless of the frequency assignment.

If the set of time slots, γ_i, represents a unique color, then the whole network can be scheduled using at most four distinct colors such that no two adjacent cells have the same color, i.e., four pair-wise disjoint sets of time slots, as shown in Fig. 5(b). Thus, the total number of time slots required is 4 times the maximum number of slots in any set γ_i. ■

LEMMA 5: If L_i^{ϕ} denote the load on the maximally loaded frequency in cell c_i under mapping $\phi : \mathcal{R}_i \rightarrow \{f_1, \ldots, f_K\}$ achieved by algorithm FREQUENCY-GREEDY, then any greedy time slot assignment scheme can schedule all the edges in cell c_i within $2L_i^{\phi}$ time slots.

Proof: Consider a multi-graph $H = (\{f_1, \ldots, f_K\}, E')$, where for each edge $e = (v_i, v_{i'})$, $v_i, v_{i'} \in \mathcal{R}_i$ with $\phi(v_i) \neq \phi(v_{i'})$, we have an edge $(\phi(v_i), \phi(v_{i'})) \in E'$. Note that these will be multi-edges; let $n(f_k, f_{k'})$ denote the number of edges between f_k and $f_{k'}$ in H. Then, $deg(f_k) \leq l_i^{\phi}(f_k)$, where $l_i^{\phi}(f_k)$ is the load on f_k under ϕ in cell c_i. By Ore's theorem [18], which generalizes Vizing's theorem for edge coloring on multi-graphs, it follows that the edges in H can be colored using $\max_k\{l_i^{\phi}(f_k)\}$ colors. Therefore, all edges of the form $e = (v_i, v_{i'})$ between two nodes in $\mathcal{R}_i$ with different frequencies can be colored in $\max_k\{l_i^{\phi}(f_k)\} = L_i^{\phi}$ colors.

All the remaining edges either have only one end-point in $\mathcal{R}_i$, or have both end-points in $\mathcal{R}_i$, with the same frequency on their receivers; let $S(f_k)$ denote the set of such edges with their end-point in $\mathcal{R}_i$ that are assigned frequency f_k. Note that $|S(f_k)| \leq l_i^{\phi}(f_k)$, and edges $e \in S(f_k), e' \in S(f_{k'})$ can be assigned the same time slot if $f_k \neq f_{k'}$. So all the remaining edges can be scheduled in $\max_k |S(f_k)| \leq \max_k\{l_i^{\phi}(f_k)\}$ time slots. Therefore, all edges in c_i can be scheduled within $2 \cdot \max_k\{l_i^{\phi}(f_k)\} = 2L_i^{\phi}$ time slots, and the lemma follows. ■

We now prove our key approximation result.

THEOREM 3: Given a routing tree T on an *arbitrarily* deployed network in 2-D, and K orthogonal frequencies, there exists a greedy algorithm $\mathcal{A}$ that achieves a constant factor $8\mu_\alpha \cdot \left(\frac{4}{3} - \frac{1}{3K}\right)$-approximation on the optimal schedule length, where $\mu_\alpha > 0$ is a constant for any $\alpha \geq 2\eta R$.

Proof: Algorithm $\mathcal{A}$ consists of two phases. In Phase 1, we run algorithm FREQUENCY-GREEDY to assign the K frequencies to the receivers in each cell. In Phase 2, we greedily schedule a *maximal* number of edges in each time slot. Let the schedule length of algorithm $\mathcal{A}$ be $\Gamma_{\mathcal{A}}$, and that of an optimal algorithm be OPT.

Due to the presence of interfering links, there exists a constant $\mu_\alpha > 0$ depending on cell size α, such that at most μ_α edges in any cell, whose receivers are on the same frequency, can be scheduled in the same time slot by an optimal algorithm.

Now, regardless of the assignment chosen by an optimal strategy, it will take at least $\max_i\{L_i^{m^*}/\mu_\alpha\}$ time slots to schedule all the edges. This is because $L_i^{m^*}$ is the *minimum* of the *maximum* number of edges that are on the same frequency in cell c_i. Thus,

$$OPT \geq \frac{1}{\mu_\alpha} \cdot \max_i \left\{ L_i^{m^*} \right\} \tag{3}$$

By running FREQUENCYGREEDY in cell c_i, Lemma 3 implies

$$L_i^\phi \leq \left(\frac{4}{3} - \frac{1}{3K} \right) \cdot L_i^{m^*}, \tag{4}$$

and by scheduling a maximal number of edges in each time slot, Lemma 5 implies $|\gamma_i| \leq 2L_i^\phi$. Then, from Lemma 4:

$$\begin{aligned} \Gamma_{\mathcal{A}} &\leq 4 \cdot \max_i \{|\gamma_i|\} \\ &\leq 8 \cdot \max_i \left\{ L_i^\phi \right\} \\ &\leq 8 \cdot \max_i \left\{ \left(\frac{4}{3} - \frac{1}{3K} \right) \cdot L_i^{m^*} \right\} \\ &\leq 8\mu_\alpha \cdot \left(\frac{4}{3} - \frac{1}{3K} \right) \cdot OPT \end{aligned}$$

■

We now derive a bound for μ_α using a classical result of circle packing due to Groemer [34]. Under the protocol interference model, two edges can transmit simultaneously if their receivers are at least a distance ηR away from the non-intended transmitters. This implies that the maximum number of edges μ_α that can be scheduled simultaneously (on the same frequency) within a single cell is upper bounded by the maximum number of nodes that can be placed with mutual distance at least ηR. From Groemer's inequality, we know that for a compact, convex set C, the number of points of mutual distance at least 1 is bounded by $\frac{2A(C)}{\sqrt{3}} + \frac{P(C)}{2} + 1$, where $A(C)$ is the area and $P(C)$ is the perimeter of C. For a square grid cell of size α, this equates to $\frac{2\alpha^2}{\sqrt{3}} + 2\alpha + 1$, and thus is an upper bound for μ_α assuming $\eta R \geq 1$.

V. ROUTING TREE CONSTRUCTION

We now turn our attention to the routing tree construction problem, where our goal is to design an (α, β)-bicriteria approximation to compute a *Bounded-Degree-Minimum-Radius Spanning Tree*, such that the radius of the tree is at most β times the minimum possible radius for a given degree bound Δ^*, and the degree of any node is at most $\Delta^* + \alpha$, where α and β are positive constants.

A. A Bicriteria Approximation Algorithm

We tessellate the 2-D deployment region into a set of hexagonal grid cells each of side length $R/2$, as shown in Fig. 6. We associate each node to a unique cell whose center is closest to the node, breaking ties arbitrarily. We define a cell to be *non-empty* if it has at least one node, and define two cells to be *neighbors* of each other if they share at least one common side. The basic idea of the spanning tree construction algorithm is as follows.

The algorithm runs in two phases. In Phase 1, we construct a *backbone tree*, $T_B = (V_B \subseteq V, E_B \subseteq E)$, from the original graph G by choosing one representative node, called *local root*, arbitrarily from each non-empty cell and connecting them in a BFS order starting from the sink. While constructing T_B, we also ensure that the hop distances along it are not too long compared to a shortest-path tree on G. This backbone tree determines the global structure of our solution.

In Phase 2, we construct a *local spanning tree* of minimum radius within each cell from the remaining nodes in $V \setminus V_B$ lying in that cell, while respecting the degree bound Δ^*. This is always possible because the nodes within each cell form a *complete graph*, as the diameter of the circumcircle for each hexagonal cell is R. Finally, we construct the overall spanning tree T by taking the union of the backbone tree and all the local spanning trees. During the execution of the algorithm, we *mark* a cell if its local root has been included in T_B; otherwise the cell is unmarked. A formal description of the algorithm is given in Algorithm 3. We now describe the two phases in detail below.

Phase 1 - Backbone Tree Construction:

1) In the beginning, all the cells are unmarked. We initialize T_B with the sink s and mark its cell c_s.
2) Choose one local root arbitrarily from each non-empty cell. Let this set of nodes be $\mathcal{R} = \{r_1, \ldots, r_n\}$.
3) Consider those unmarked adjacent cells $\{c_j\}$ of s which intersect a circle of radius R centered at s, and for which one of the following conditions is met.
   a) Local root r_j in cell c_j is a direct neighbor of s.
   b) Local root r_j in cell c_j is not a direct neighbor of s, but there exists some other node w_k, called a *helper node*, that is a common neighbor of both s and r_j.
   c) Local root r_j in cell c_j is neither a direct neighbor of s nor there is any helper node, but there exists a *helper edge* $(w_k, w_{k'})$ whose one end, say w_k, is incident in cell c_j and the other end $w_{k'}$ in cell c_s.

   In Fig. 6, these are the shaded cells.
4) For case a), connect r_j directly to s and mark its cell c_j. Update T_B by adding r_j to V_B, and the edge (r_j, s) to E_B. Nodes r_2 and r_6 in Fig. 6 are such nodes.
5) For case b), connect r_j to s via the helper node. Mark c_j and update T_B by adding r_j and w_k to V_B, and the two edges (r_j, w_k) and (w_k, s) to E_B. In Fig. 6, nodes r_3 and r_5 are connected to s via w_1, and node r_4 via w_2.
6) For case c), connect r_j to s via the helper edge. Mark c_j and update T_B by adding r_j, w_k, and $w_{k'}$ to V_B, and the three edges (r_j, w_k), $(w_k, w_{k'})$, and $(w_{k'}, s)$ to E_B. In Fig. 6, node r_1 is connected via $(w_1, w_{1'})$.
7) Consider, in BFS order, these marked cells $\{c_j\}$ and repeat steps 3-6 with node s replaced by the corresponding local root in c_j.
8) Continue until all the local roots in $\mathcal{R}$ get connected.

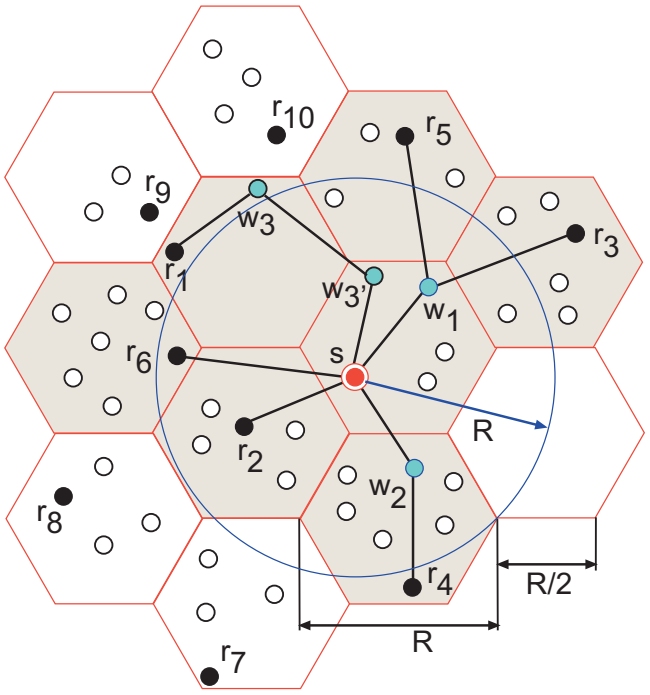


Fig. 6. Backbone tree construction: Filled black circles represent local roots (chosen arbitrarily from each non-empty cell), and shaded cells are adjacent cells of s that intersect the circle of radius R centered at s, and satisfy one of the conditions (a), (b), or (c) of Phase 1. Iteration 1: Local roots r_1, r_2, r_3, r_4, r_5, and r_6 are connected to s. Nodes r_3 and r_5 are connected to s via helper node w_1, and node r_4 via helper node w_2; node r_1 is connected via helper edge $(w_3, w_{3'})$.

We implement the BFS processing of the local roots in a queue data structure.

Phase 2 - Local Spanning Tree Construction:

Consider the local root r_j in cell c_j. Let the set of nodes in c_j that are not yet connected to the backbone tree be $V_j = \{v_1, \ldots, v_{n_j}\} \subset V \setminus V_B$. Connect v_1 to r_j treating r_j as its parent. Then, connect at most $\Delta^* - 1$ nodes (if those many exist) from V_j to v_1; these constitute the direct neighbors of v_1. Next, treating these direct neighbors as parents, connect at most $\Delta^* - 1$ nodes to each one of them, if those many exist. Fig. 7(a) shows an illustration of this phase. Continue this until there is no isolated node left in V_j, and repeat the procedure for each of the non-empty cells. At the end of this phase, each c_j contains a local spanning tree T_j rooted at r_j, with each node (except the leaves and the last parent) having degree Δ^*. The overall spanning tree T is the union of the backbone tree T_B and all the local spanning trees T_j.

Algorithm 3 Approximation algorithm for *Bounded-Degree Minimum-Radius Spanning Tree*

Input: $G = (V, E)$; sink s; degree bound $\Delta^* \geq 2$
Output: BDMRST T of G
Tessellate the 2-D region into hexagonal grid cells, each of side length $R/2$.
Associate each node to a unique cell whose center is closest to the node.
Phase 1: Backbone Tree
All cells are unmarked.
Initialize T_B: $V_B \leftarrow \{s\}$, $E_B \leftarrow \phi$, mark cell of s.
Choose one local root arbitrarily from each non-empty cell; let $\mathcal{R} = \{r_1, \ldots, r_n\}$ be the set of local roots.
$\mathcal{Q} \leftarrow \phi$;
ENQUEUE($\mathcal{Q}$, s);
while $\mathcal{Q} \neq \phi$ **do**
  $u \leftarrow$ DEQUEUE($\mathcal{Q}$);
  for all unmarked cells c_j adjacent to u **do**
    $r_j \leftarrow$ local root in c_j;
    if Case (a) **then**
      $V_B \leftarrow V_B \cup \{r_j\}$;
      $E_B \leftarrow E_B \cup \{(u, r_j)\}$;
      Mark c_j;
      ENQUEUE($\mathcal{Q}$, r_j);
    else if Case (b) **then**
      $V_B \leftarrow V_B \cup \{r_j, w_k\}$;
      $E_B \leftarrow E_B \cup \{(r_j, w_k), (w_k, u)\}$;
      Mark c_j;
      ENQUEUE($\mathcal{Q}$, r_j);
    else if Case (c) **then**
      $V_B \leftarrow V_B \cup \{r_j, w_k, w_{k'}\}$;
      $E_B \leftarrow E_B \cup \{(r_j, w_k), (w_k, w_{k'}), (w_{k'}, u)\}$;
      Mark c_j;
      ENQUEUE($\mathcal{Q}$, r_j);
    end if
  end for
end while
Phase 2: Local Spanning Tree
for all non-empty cells c_j **do**
  $r_j \leftarrow$ local root in c_j;
  Let $V_j = \{v_1 \ldots v_{n_j}\}$ be the set of not yet connected nodes in c_j (V_j induces a complete graph).
  Construct local spanning tree T_j of minimum radius with nodes in V_j such that no node exceeds degree Δ^*.
end for
return $T = T_B \cup \{T_j\}$.

B. Algorithm Analysis

THEOREM 4: Algorithm 3 gives an (α, β)-bicriteria approximation to the Bounded-Degree-Minimum-Radius Spanning Tree, where $\alpha = 10$ and $\beta = 7$.

The proof unfolds in the following lemmas.

LEMMA 6: Let r_i and r_j be any two local roots on the backbone tree T_B. Let $P_G(r_i, r_j)$ be the shortest path on the original graph G between r_i and r_j consisting of m hops. Then, the length of the unique simple path $P_{T_B}(r_i, r_j)$ between r_i and r_j on T_B is at most $6m$.

Proof: We first show that all the local roots get connected to the backbone tree T_B. Since the original graph G is connected and the side length of each hexagonal cell is $R/2$, every non-empty cell must have at least one edge that crosses the boundary of another non-empty cell. This means that the local root of every non-empty cell will be able to connect to the local root of at least another non-empty cell using a helper node or at most one helper edge. Since the backbone tree is constructed precisely in this manner, i.e., by connecting a local

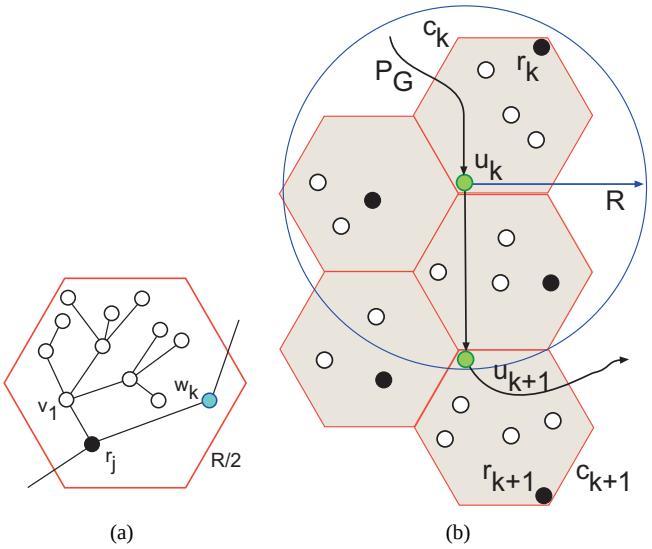


Fig. 7. (a) Local tree construction on an induced complete graph within each cell for maximum node degree $\Delta^* = 4$; filled black circle represents the local root. (b) Traversing the edge (u_k, u_{k+1}) along the shortest path P_G in graph G. Local roots r_k and r_{k+1} are at most distance $3R$ away from each other.

root either directly, or using a helper node, or a helper edge, all the local roots get connected to T_B.

Let $P_G(r_i, r_j) = \{r_i = u_0, u_1, \ldots, u_{m-1}, u_m = r_j\}$ be the shortest path on graph G, and $P_{T_B}(r_i, r_j) = \{r_i = v_0, v_1, \ldots, v_{h-1}, v_h = r_j\}$ be the unique simple path on the backbone tree T_B. Note that each v_k is either a local root or a helper node. We will show that for every edge (u_k, u_{k+1}) in P_G we add at most a constant number of (v_k, v_{k+1}) edges in P_{T_B}.

Consider the nodes $u_1, \ldots, u_{m-1}$, and traverse them in the order as they appear in P_G. In parallel, traverse the path in P_{T_B} by *tracking* the progress in P_G, i.e., for every edge traversed in P_G, we traverse a certain number of edges in P_{T_B}. Both traversals start from the same cell (and the same node r_i). Suppose at any give point, we are about to traverse the edge (u_k, u_{k+1}). One of the two possibilities could occur: (i) u_k and u_{k+1} lie in the same cell, or (ii) u_k and u_{k+1} lie in different cells, say c_k and c_{k+1}, respectively, as shown in Fig. 7(b). In the first case, we do not traverse any edge in P_{T_B}. In the second case, we make our traversal along the local roots and helper nodes in P_{T_B}, such that the end point of the last edge traversed in P_{T_B} is a local root r_{k+1} that lies in the same cell as u_{k+1}. This is always possible because each non-empty cell contains a local root.

Since the Euclidean length of the edge (u_k, u_{k+1}) is at most R, u_{k+1} must lie in one of the adjacent cells of u_k. Also, since the side length of each cell is $R/2$, the distance between the local roots r_k and r_{k+1} lying in cells c_k and c_{k+1}, respectively, is at most $3R$. Now, during the backbone tree construction phase in Algorithm 3, we first connect the local roots from *all* the non-empty adjacent cells of an already connected local root before exploring other cells. Since connecting a local root to the backbone tree takes at most 3 edges (when helper edges are needed), the number of edges traversed on P_{T_B} for case (ii) is at most 6. Therefore, the length of the path P_{T_B} is at most 6 times the length of the path P_G. ■

LEMMA 7: The degree of a local root, a helper node, or a node on a helper edge in the backbone tree is at most 12.

Proof: Recall that during the backbone tree construction, a new local root from each non-empty cell adjacent to an already chosen local root r_j is connected either directly, via a helper node, or via a helper edge to r_j. This increases the degree of r_j by at most one. Since the side length of each cell is $R/2$, the maximum number of cells that are adjacent to r_j is at most 11; this will happen when r_j lies near one of the corners within its cell c_j. There will be 6 cells that are neighbors to c_j (i.e., share exactly one side with c_j) and 5 other cells that are not neighbors to c_j. Also, recall that during constructing the local spanning trees within each cell, at most one node is connected to the local root of that cell. Therefore, the degree of any local root in the backbone tree is at most 12 in the worst case. A similar argument also holds for the degree of any helper node. ■

Proof: (Theorem 4) (*Bound on the Radius*): Let the radius of an optimal spanning tree whose maximum node degree is Δ^* on graph G be OPT, and let that produced by Algorithm 3 be $R(T)$. Suppose v^* is a node farthest from sink s, and let m be the hop distance on a shortest path tree (no degree bound) from s to v^* on graph G. Then, $OPT \geq m$, because a degree constraint can only increase the radius of a tree.

Now the node v^* can either be a local root, or a helper node, or a node in a local spanning tree.

- if v^* is a local root, then by Lemma 6, $R(T) \leq 6m \leq 6 \cdot OPT$.
- if v^* is a helper node or a node on a helper edge, then the shortest path on the tree comprises a path from s to the local root of the cell containing v^*, plus an additional hop from the local root to v^*. Thus, $R(T) \leq 6m + 1 \leq 6 \cdot OPT + 1$.
- if v^* is a node in a local spanning tree T_j, then the shortest path on the tree comprises a path from s to the local root of the cell containing v^*, plus at most $R(T_j)$ hops from the local root to v^*. Thus,

$$\begin{aligned} R(T) &\leq 6m + R(T_j) \\ &\leq 6 \cdot OPT + OPT \\ &= 7 \cdot OPT \end{aligned}$$

The second inequality follows because the radius of each local spanning tree T_j constructed on the complete graph within each cell is minimum, respecting degree constraint Δ^*, and so $OPT \geq R(T_j)$.

(*Bound on the Degree*): From Lemma 7, the maximum node degree of any node in the backbone tree is at most 12. Also, the degree of any node is a local spanning tree is at most Δ^*. Thus, the degree of any node in the overall spanning tree is bounded by $\max(\Delta^*, 12) \leq \Delta^* + 10$, for any $\Delta^* \geq 2$.

Therefore, the theorem follows with $\alpha = 10$ and $\beta = 7$. ■

VI. EVALUATION

In this section, we evaluate the performance of our scheduling and routing tree construction algorithms using Matlab simulations on networks modeled as *random geometric graphs*.

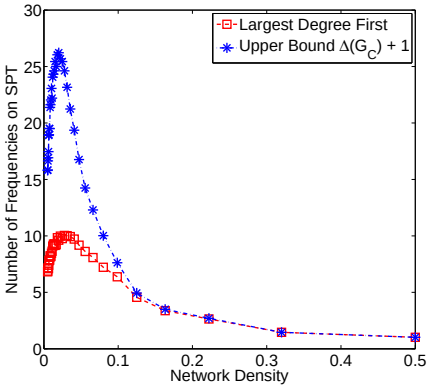


Fig. 8. Number of frequencies required to remove all the secondary conflicts as a function of network density on shortest path trees.

We generate connected networks by uniformly and randomly placing nodes in a square region of maximum size 200×200, and connecting any two nodes that are at most distance $R = 25$ apart.

A. Frequency Bounds

In Fig. 8, we compare the number of frequencies needed as a function of network density to remove all the secondary conflicts on shortest path trees, as calculated from the upper bound $\Delta(G_C) + 1$, and that from *Largest Degree First* (LDF) assignment. Here, the number of nodes N is fixed at 200, and the length l of the square region is varied from 200 to 20; so the density, $d = N/l^2$, varies from 0.005 to 0.5.

The plot shows that the number of frequencies initially increases with density, reaching a peak at around 0.025, and then steadily going down to one. This happens because of two opposing factors. As the density goes up, the parents link up with more and more new nodes, increasing the number of secondary conflicts; however, at the same time, the number of parents on the SPT gradually decreases because the deployment region gets smaller in size. As we keep on increasing the density further, the latter effect starts dominating, and since the number of frequencies required depends on the number of parents in the constraint graph G_C, this number goes down as well, until the network finally turns into a single hop network with the sink as the only parent. We also observe that for sparser networks there is a significant gap between the upper bound and the LDF scheme, as opposed to that in denser networks. This is because in sparser settings there are many parents, resulting in a higher $\Delta(G_C)$ value, and assigning a distinct frequency to the largest degree parent according to the LDF scheme removes more secondary conflicts at every step than it does for denser settings when the parents are fewer and have comparable degrees.

B. Schedule Length and Maximum Delay

We evaluate the performance of the multi-channel scheduling algorithm $\mathcal{A}$ of Theorem 3 on three different kinds of spanning trees – BDMRST, SPT, and MIT – with the size deployment region fixed at 200×200. Note that, the constant approximation factor in our algorithm depends on the parameter μ_α. Since μ_α decreases with decreasing α, and the smallest α for which Lemma 4 holds is $2R$, we choose $\alpha = 50$. We first present the results for a single frequency, and then discuss the case with multiple frequencies.

Fig. 9(a) and 9(b) show the schedule length and the maximum packet delay, respectively, with increasing network size on three different types of trees for a single frequency. Each point in the plot is averaged over 20 iterations. In each iteration, we deploy the nodes uniformly and randomly, construct the spanning trees, and then run the scheduling algorithm. In other words, we keep the node deployment fixed in a given iteration for all the three tree types in order to preserve the underlying communication graph and minimize any statistical variation. The maximum degree bound on the BDMRST is taken as 4. We observe that the schedule lengths on a BDMRST and MIT are very close to each other, whereas those on an SPT are much higher. This difference becomes more predominant with increasing network size because the maximum node degrees on an SPT go up rapidly, as shown in Fig. 9(c). On the other hand, the maximum packet delays on an SPT are minimum, and those on a BDMRST are very close. However, the delays on an MIT are much higher due to its very small and almost constant node degrees throughout, as shown in Fig. 9(c), which give rise to longer hop distances. Thus, scheduling on a BDMRST achieves the best of both worlds in terms of having a small schedule length as well as very close to smallest possible maximum delay.

C. Multiple Frequencies on Schedule Length

Since multiple frequencies can eliminate interfering links and reduce the schedule length, we now evaluate their effects on three different kinds of trees for our proposed multi-channel scheduling algorithm. Fig. 10(a), 10(b), and 10(c) show the schedule lengths with increasing network size for one, three, and five frequencies on SPT, MIT, and BDMRST, respectively. We observe that with SPT and MIT, the gains of utilizing multiple frequencies increase as the network gets larger in size. In the case of an SPT, the schedule lengths with three and five frequencies are almost the same. This is due to very high node degrees on an SPT resulting in many primary conflicts in the network than secondary conflicts. Recall that primary conflicts are not removable using multiple frequencies.

We also observe that an MIT benefits the most with multiple frequencies. This is because an MIT has small node degrees and very large hop distances to the sink (cf. Fig. 1(b)), which gives rise to a lot of secondary conflicts but only a very few primary conflicts. A typical path on an MIT from any node to the sink looks almost like a linear network, where every non-adjacent edge can be scheduled simultaneously with two frequencies. Note that, with one frequency, only *distance-2 edges*, i.e., edges whose end points are not incident on a common edge, can be scheduled simultaneously. Lastly, we see that the schedule lengths on a BDMRST do not improve at all with multiple frequencies. This is due to almost constant maximum hop distances to the sink and nearly constant maximum node degrees, as shown in Fig. 9(b) and 9(c).

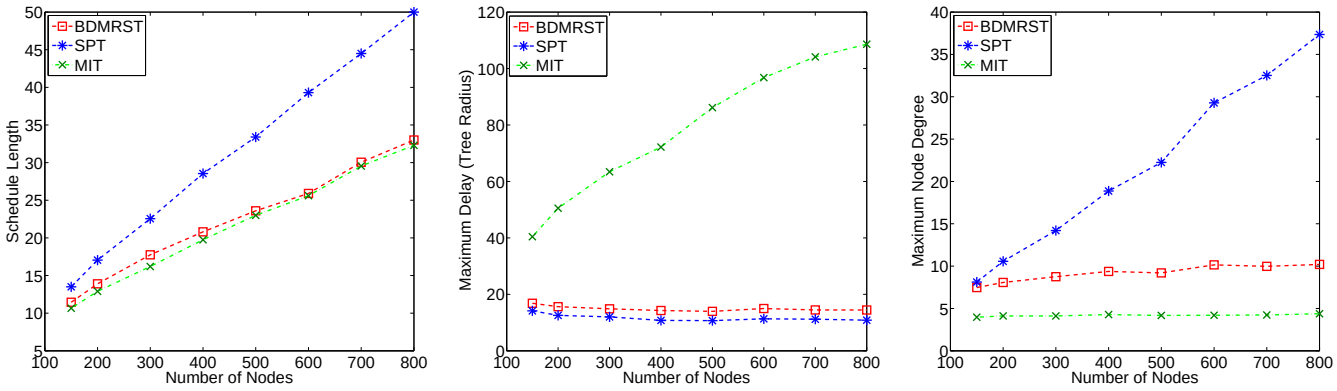


Fig. 9. (a) Schedule Length, (b) Maximum Delay (tree radius), and (c) Maximum Node Degree with increasing network size on three different types of trees (BDMRST, SPT, and MIT) for single frequency scheduling.

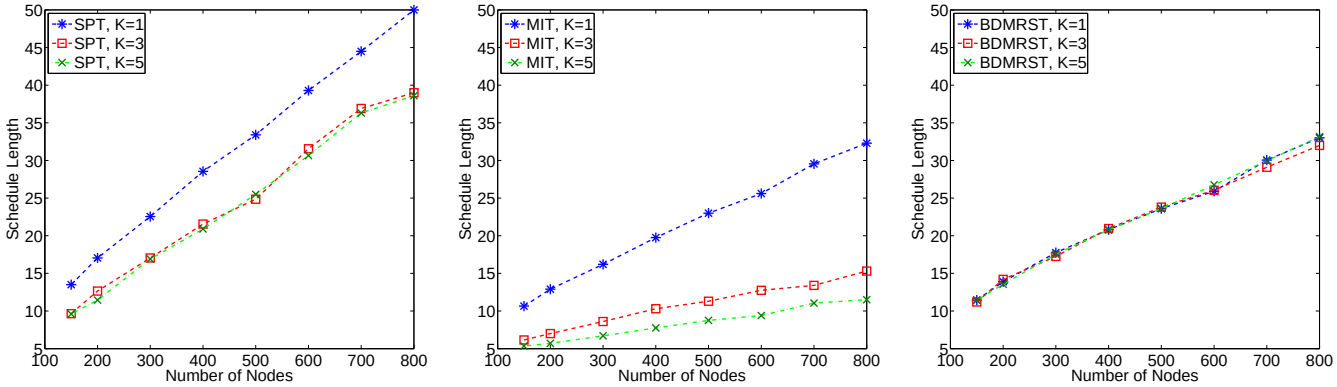


Fig. 10. Effect of multiple frequencies: Schedule Lengths for (a) SPT, (b) MIT, and (c) BDMRST with network size for $K = 1$, 3, and 5 frequencies.

D. Degree Distribution and SINR Model

We note that the maximum node degrees on an SPT are very high compared to those on an MIT and BDMRST. Furthermore, they are nearly constant throughout the network size for both MIT and BDMRST. In order to gain more insights on the effects of node degrees on the schedule length, we plot the degree distribution of BDMRST, SPT, and MIT for three different network sizes with $N = 150$, 500, and 800 nodes in Fig. 11(a), 11(b), and 11(c), respectively. The bar graphs show the number of nodes that have degrees of particular values averaged over 20 iterations for each tree type.

For all network sizes, we observe that most of the nodes on an SPT have degree one, whereas few have degrees very high. This is because large groups of degree one nodes (leaf nodes) are connected to common parents, giving rise to a lot of primary conflicts, and thereby being more resistant to improving the schedule length with multiple frequencies. In MIT, we see that most of the nodes have degree two, and no node has degree more than five. This is because most of the paths from any node to the sink on an MIT look like a linear topology. We also observe that an MIT has a lot of parent nodes compared to an SPT, thus further explaining the reason for much more improvement in the schedule length with multiple frequencies. Lastly, the degrees on a BDMRST are more evenly distributed for all network sizes, as illustrated in Fig. 12 by a sample tree constructed on the same deployment of 800 nodes of Fig. 1(a).

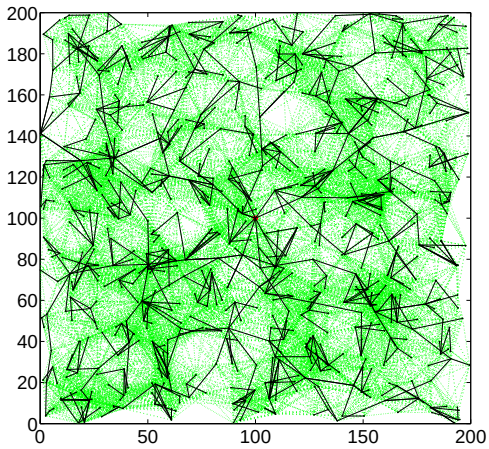


Fig. 12. A BDMRST constructed on the same deployment of 800 nodes of Fig. 1(a) and 1(b). The node degrees are more uniform compared to those on an SPT and MIT.

In our evaluation so far, we have considered the graph-based protocol interference model and evaluated the proposed scheduling algorithm. However, since the protocol model devi-

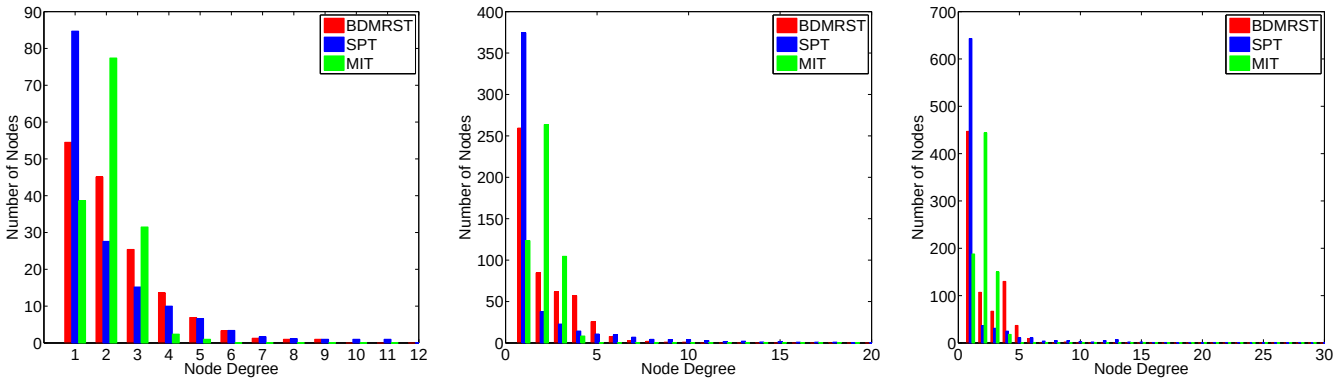


Fig. 11. Node Degree Distribution of BDMRST, SPT, and MIT for three different network sizes with (a) $N = 150$, (b) $N = 500$, and (c) $N = 800$ nodes.

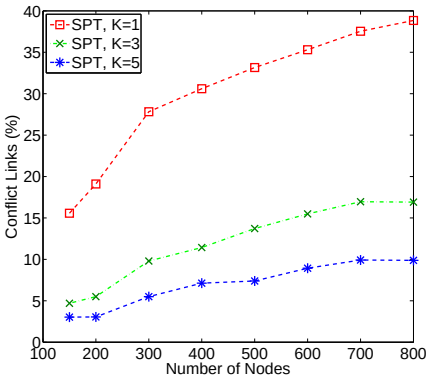


Fig. 13. Percentage of nodes whose schedules conflict in the SINR model for different network sizes and three different number of frequencies ($K = 1$, 3, 5) on an SPT.

ates from the more realistic *Signal-to-Interference-plus-Noise-Ratio* (SINR) model in capturing cumulative interference from far-away transmitters, thus sometimes under/over estimating interference, the schedules generated from the protocol model might conflict under the SINR model. To measure the amount of conflict, we plot in Fig. 13 the percentage of nodes on an SPT whose schedules calculated according to the protocol model conflict under the SINR model. The parameters for the SINR model are chosen according to the CC2420 radio parameters with receiver sensitivity -95 dB, path-loss exponent 3.5, and transmit power -6 dB. We note that for a given number of frequencies, as the network gets denser the amount of conflict increases, reaching almost 40% for the densest deployment; however, with multiple frequencies, the amount of conflict is much less. This indicates that although the protocol model performs reasonably well under multiple frequencies, more sophisticated SINR based scheduling algorithms are needed when there is only one frequency.

VII. Conclusions and Future Work

In this paper, we discussed the trade-off between aggregated sink throughput and packet delays for fast data collection in sensor networks. We showed that multiple frequencies and bounded-degree minimum-radius spanning trees can help in achieving the best of both worlds in terms of maximizing the throughput as well as minimizing the maximum packet delay. To this end, we proposed a multi-channel scheduling algorithm that has a worst-case constant factor approximation guarantee on the schedule length for arbitrarily deployed networks in 2-D. We also designed a spanning tree construction algorithm that achieves a constant factor bicriteria approximation guarantee on minimizing the maximum hop distance in the tree under a given node degree constraint. Our future work lies in extending the multi-channel scheduling algorithm for the more realistic SINR model in order to capture the cumulative interference from concurrently transmitting distant nodes. We also want to explore transmission power control mechanisms on the nodes to save energy. To this end, considering general disk graphs where nodes can have different transmission ranges is also part of our future work.

References

[1] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, “A Wireless Sensor Network for Structural Monitoring”, in *SENSYS ’04*, pp. 13-24.

[2] Wildfire Detection, Crossbow Technology, http://www.xbow.com/Eko/EnvironmentalWildfire Detection.aspx

[3] J. Beutel, S. Gruber, A. Hasler, R. Lim, A. Meier, C. Plessl, I. Talzi, L. Thiele, C. Tschudin, M. Woehrle, and M. Yuecel, “PermaDAQ: A Scientific Instrument for Precision Sensing and Data Recovery in Environmental Extremes” in *IPSN ’09*, pp. 265–276.

[4] M. Burkhart, P. von Rickenbach, R. Wattenhofer, and A. Zollinger, “Does topology Control Reduce Interference”, in *MOBIHOC ’04*, pp. 9–19.

[5] V. Annamalai, S. Gupta, and L. Schwiebert, “On Tree-Based Convergecasting in Wireless Sensor Networks”, in *WCNC ’03*, pp. 1942-1947.

[6] X. Chen, X. Hu, and J. Zhu, “Improved Algorithm for Minimum Data Aggregation Time Problem in Wireless Sensor Networks”, in *Journal of Systems Science and Complexity*, 21(4):626–636, 2005.

[7] T. Moscibroda, “The Worst-Case Capacity of Wireless Sensor Networks”, in *IPSN ’07*, pp. 1–10.

[8] Ö. D. Incel and B. Krishnamachari “Enhancing the Data Collection Rate of Tree-Based Aggregation in Wireless Sensor Networks”, in *SECON ’08*, pp. 569–577.

[9] A. Ghosh, Ö. D. Incel, V. S. Anil Kumar, and B. Krishnamachari, “Multi-Channel Scheduling Algorithms for Fast Aggregated Convergecast in Sensor Networks”, in *MASS ’09*, pp. 363–372.

[10] H. Choi, J. Wang, and E. A. Hughes, "Scheduling on Sensor Hybrid Network", in *ICCCN '05*, pp. 505–508.
[11] N. Lai, C. King, and C. Lin, "On Maximizing the Throughput of Convergecast in Wireless Sensor Networks", in *GPC '08*, pp. 396-408.
[12] S. Gandham, Y. Zhang, and Q. Huang, "Distributed Time-Optimal Scheduling for Convergecast in Wireless Sensor Networks", in *Computer Networks*, 52(3):610-629, 2008.
[13] J. So and N. H. Vaidya, "Multi-Channel MAC for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals using a Single Transceiver", in *MOBIHOC '04*, pp. 222–233.
[14] P. Bahl, R. Chandra, and J. Dunagan, "SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks", in *MOBICOM '04*, pp. 216–230.
[15] G. Zhou, C. Huang, T. Yan, T. He, J.A. Stankovic, and T. F. Abdelzaher, "MMSN: Multi-Frequency Media Access Control for Wireless Sensor Networks", in *INFOCOM '06*, pp. 1–13.
[16] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks", in *OSDI '02*, pp. 131–146.
[17] A. Ghosh, Ö. D. Incel, V. S. Anil Kumar, and B. Krishnamachari, "Algorithms for Fast Aggregated Convergecast in Sensor Networks", *USC CENG Tech. Report, CENG-2008-8*.
[18] O. Ore, "The Four-Color Problem", *Academic Press*, 1967.
[19] R. L. Graham, "Bounds on Multiprocessing Timing Anomalies", in *SIAM Journal on Applied Mathematics*, 17(2):416–429, 1969.
[20] M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", *W. H. Freeman & Company*, 1979.
[21] C. Zhou and B. Krishnamachari, "Localized Topology Generation Mechanisms for Self-Configuring Sensor Networks", in *GLOBECOM '03*, pp. 1269–1273.
[22] M. Furer and B. Raghavachari, "Approximating the Minimum Degree Spanning Tree to Within One of the Optimal Degree", in *SODA '92*, pp. 317–324.
[23] M. Singh and L. C. Lau, "Approximating Minimum Bounded Degree Spanning Trees to Within One of Optimal", in *STOC '07*, pp. 661–670.
[24] J. M. Ho, D. T. Lee, and C.-H. Chang, "Bounded-Diameter Minimum Spanning Trees and Related Problems", in *SCG '89*, pp. 276-282.
[25] R. Hassin and A. Tamir, "On the Minimum Diameter Spanning Tree Problem", in *Information Processing Letters*, 53(2):109–111, 1995.
[26] R. Ravi, "Rapid Rumor Ramification: Approximating the Minimum Broadcast Time", in *FOCS '94*, pp. 202–213.
[27] J. Könemann, A. Levin, and A. Sinha, "Approximating the Degree-Bounded Minimum Diameter Spanning Tree Problem", in *Algorithmica*, 41(2):117–129, 2005.
[28] R. Ravi, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt, III, "Many Birds with One Stone: Multi-Objective Approximation Algorithms", in *STOC '93*, pp. 438–447.
[29] A. Raniwala and C. Tzi-cker, "Architecture and Algorithms for an IEEE 802.11-based Multi-channel Wireless Mesh Network", in *INFOCOM '05*, pp. 2223–2234.
[30] Y. Wu, J. A. Stankovic, T. He, and S. Lin, "Realistic and Efficient Multi-Channel Communications in Wireless Sensor Networks", in *INFOCOM '08*, pp. 1193–1201.
[31] X. Lin and S. Rasool, "A Distributed Joint Channel Assignment, Scheduling and Routing Algorithm for Multi-Channel Ad-Hoc Wireless Networks", in *INFOCOM '07*, pp. 1118-1126.
[32] B. Raman, K. Chebrolu, S. Bijwe, and V. Gabale, "PIP: A Connection-Oriented, Multi-Hop, Multi-Channel TDMA-based MAC for High Throughput Bulk Transfer", in *SenSys '10*, pp. 15–28.
[33] D. Hochbaum and D. B. Shmoys, "Using dual approximation algorithms for scheduling problems: theoretical and practical results", in *J. ACM*, 34(1):144–162, 1987.
[34] H. Groemer, "Über die Einlagerung von Kreisen in einen konvexen Bereich", in *Mathematische Z.*, 73(3):285–294, 1960.
[35] Ö. D. Incel, A. Ghosh, B. Krishnamachari, and K. K. Chintalapudi, "Fast Data Collection in Tree-Based Wireless Sensor Networks", *USC CENG Tech. Report, CENG-2010-8*.

Amitabha Ghosh is currently a postdoctoral research associate in Princeton University. He received his Ph.D. in Electrical Engineering from the University of Southern California in August 2010. His primary research interest is in the design and analysis of algorithms for scheduling, routing, and power control for wireless networks. He received his B.S. in Physics from the Indian Institute of Technology, Kharagpur, in 1996, and his M.E. in Computer Science and Engineering from the Indian Institute of Science, Bangalore, in 2000. Between 2000 and 2004, he worked for Alcatel-Lucent and Honeywell on GSM/GPRS networks and wireless ad hoc networks.

Özlem Durmaz Incel is currently a postdoctoral researcher in the Networking Laboratory (NETLAB) of the Bogazici University, Turkey. She received her PhD in computer science from the University of Twente, Netherlands, in March 2009. Her dissertation focused on efficient data collection in wireless sensor networks and was entitled as Multi-Channel Wireless Sensor Networks: Protocols, Design and Evaluation. She was a visiting student in the Autonomous Networks Research Group of the University of Southern California as part of her Phd studies in 20072008. She received both her MSc and BSc degrees in computer engineering from the Yeditepe University, Turkey, in 2005 and 2002.

V. S. Anil Kumar is currently an Assistant Professor in the Department of Computer Science and the Virginia Bioinformatics Institute at Virginia Tech. His interests are in the broad areas of algorithms, combinatorial optimization, probabilistic techniques and distributed computing, and their applications to wireless networks, epidemiology, and the modeling, simulation and analysis of social and infrastructure networks. He received his Ph.D. in Computer Science from the Indian Institute of Science in 1999 and was a postdoctoral associate at the Max-Planck Institute and at Los Alamos National Laboratory.

Bhaskar Krishnamachari received his B.E. in Electrical Engineering at The Cooper Union, New York, in 1998, and his M.S. and Ph.D. degrees from Cornell University in 1999 and 2002 respectively. He is currently an Associate Professor and a Ming Hsieh Faculty Fellow in the Department of Electrical Engineering at the University of Southern California's Viterbi School of Engineering. His primary research interest is in the design and analysis of algorithms and protocols for next-generation wireless networks.