Fast Data Collection in Tree-Based Wireless Sensor Networks

Özlem Durmaz Incel, Amitabha Ghosh, Bhaskar Krishnamachari, and Krishnakant Chintalapudi

Abstract—We investigate the following fundamental question - *how fast can information be collected from a wireless sensor network organized as tree?* To address this, we explore and evaluate a number of different techniques using realistic simulation models under the many-to-one communication paradigm known as *convergecast*. We first consider time scheduling on a single frequency channel with the aim of minimizing the number of time slots required (*schedule length*) to complete a convergecast. Next, we combine scheduling with transmission power control to mitigate the effects of interference, and show that while power control helps in reducing the schedule length under a single frequency, scheduling transmissions using multiple frequencies is more efficient. We give lower bounds on the schedule length when interference is completely eliminated, and propose algorithms that achieve these bounds. We also evaluate the performance of various channel assignment methods and find empirically that for moderate size networks of about 100 nodes, the use of multi-frequency scheduling can suffice to eliminate most of the interference. Then, the data collection rate no longer remains limited by interference but by the topology of the routing tree. To this end, we construct degree-constrained spanning trees and capacitated minimal spanning trees, and show significant improvement in scheduling performance over different deployment densities. Lastly, we evaluate the impact of different interference and channel models on the schedule length.

Index Terms—Convergecast, TDMA scheduling, multiple channels, power-control, routing trees.

1 Introduction

Convergecast, namely the collection of data from a set of sensors toward a common sink over a tree-based routing topology, is a fundamental operation in wireless sensor networks (WSN) [1]. In many applications, it is crucial to provide a guarantee on the delivery time as well as increase the rate of such data collection. For instance, in safety and mission-critical applications where sensor nodes are deployed to detect oil/gas leak or structural damage, the actuators and controllers need to receive data from all the sensors within a specific deadline [2], failure of which might lead to unpredictable and catastrophic events. This falls under the category of one-shot data collection. On the other hand, applications such as permafrost monitoring [3] require periodic and fast data delivery over long periods of time, which falls under the category of continuous data collection.

In this paper, we consider such applications and focus on the following fundamental question: "*How fast can data be streamed from a set of sensors to a sink over a tree-based topology?*" We study two types of data collection: (i) *aggregated convergecast* where packets are aggregated at each hop, and (ii) *raw-data convergecast* where packets are individually relayed toward the sink. Aggregated convergecast is applicable when a strong spatial correlation exists in the data, or the goal is to collect summarized information such as the maximum sensor reading. Raw-data convergecast, on the other hand, is applicable when every sensor reading is equally important, or the correlation is minimal. We study aggregated convergecast in the context of continuous data collection, and raw-data convergecast for one-shot data collection. These two types correspond to two extreme cases of data collection. In an earlier work [4], the problem of applying different aggregation factors, i.e., data compression factors, was studied, and the latency of data collection was shown to be within the performance bounds of the two extreme cases of no data compression (raw-data convergecast) and full data compression (aggregated convergecast).

For periodic traffic, it is well known that contention-free medium access control (MAC) protocols such as TDMA (Time Division Multiple Access) are better fit for fast data collection, since they can eliminate collisions and retransmissions and provide guarantee on the completion time as opposed to contention-based protocols [1]. However, the problem of constructing conflict-free (interference-free) TDMA schedules even under the simple graph-based interference model has been proved to be NP-complete. In this work, we consider a TDMA framework and design polynomial-time heuristics to minimize the schedule length for both types of convergecast. We also find lower bounds on the achievable schedule lengths and compare the performance of our heuristics with these bounds.

We start by identifying the primary limiting factors of fast data collection, which are: (i) *interference* in the wireless medium, (ii) *half-duplex* transceivers on the sen-

- *O.D. Incel is with the Department of Computer Engineering, Bogazici University, Istanbul, Turkey, Email: ozlem.durmaz@tam.boun.edu.tr*
- *A. Ghosh is with the Department of Electrical Engineering, Princeton University, NJ, Email: amitabhg@princeton.edu*
- *B. Krishnamachari is with the Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA, Email: bkrishna@usc.edu*
- *K. Chintalapudi is with Microsoft Research, Bangalore, India.*

sor nodes, and (iii) *topology* of the network. Then, we explore a number of different techniques that provide a hierarchy of successive improvements, the simplest among which is an interference-aware, minimum-length, TDMA scheduling that enables spatial reuse. To achieve further improvement, we combine transmission power control with scheduling, and use multiple frequency channels to enable more concurrent transmissions. We show that once multiple frequencies are employed along with spatial-reuse TDMA, the data collection rate often no longer remains limited by interference but by the topology of the network. Thus, in the final step, we construct network topologies with specific properties that help in further enhancing the rate. Our primary conclusion is that, combining these different techniques can provide an order of magnitude improvement for aggregated convergecast, and a factor of two improvement for raw-data convergecast, compared to single-channel TDMA scheduling on minimum-hop routing trees.

Although the techniques of transmission power control and multi-channel scheduling have been well studied for eliminating interference in general wireless networks, their performances for bounding the completion of data collection in WSNs have not been explored in detail in the previous studies. The fundamental novelty of our approach lies in the extensive exploration of the efficiency of transmission power control and multi-channel communication on achieving fast convergecast operations in WSNs. Besides, we evaluate the impact of routing trees on fast data collection and to the best of our knowledge this has not been the topic of previous studies. As we will discuss in Section 2, some of the existing work had the objective of minimizing the completion time of convergecasts. However, none of the previous work discussed the effect of multi-channel scheduling together with the comparisons of different channel assignment techniques and the impact of routing trees and none considered the problems of aggregated and raw convergecast, which represent two extreme cases of data collection, together.

As the new concepts in this paper, we introduce polynomial-time heuristics for TDMA scheduling for both types of data collection, i.e., Algorithms 1 and 2, and prove that they do achieve the lower bound of data collection time once interference is eliminated. Besides, we elaborate on the performance of our previous work, a receiver-based channel assignment method, and compare its efficiency with other channel assignment methods and introduce heuristics for constructing optimal routing trees to further enhance data collection rate. The following lists our key findings and contributions:

- **Bounds on Convergecast Scheduling**: We show that if all interfering links are eliminated, the schedule length for aggregated convergecast is lower bounded by the maximum node degree in the routing tree, and for raw-data convergecast by $\max(2n_k - 1, N)$, where n_k is the maximum number of nodes on any branch in the tree, and N is the number of source nodes. We then introduce optimal time slot assignment schemes under this scenario which achieve these lower bounds.
- **Evaluation of Power Control under Realistic Setting**: It was shown recently [5] that under the idealized setting of unlimited power and continuous range, transmission power control can provide an unbounded improvement in the asymptotic capacity of aggregated convergecast. In this work, we evaluate the behavior of an optimal power control algorithm [6] under realistic settings considering the limited discrete power levels available in today's radios. We find that for moderate size networks of 100 nodes power control can reduce the schedule length by $15 - 20\%$.
- **Evaluation of Channel Assignment Methods**: Using extensive simulations, we show that scheduling transmissions on different frequency channels is more effective in mitigating interference as compared to transmission power control. We evaluate the performance of three different channel assignment methods: (i) *Joint Frequency and Time Slot Scheduling* (JFTSS), (ii) *Receiver-Based Channel Assignment* (RBCA) [7], and (iii) *Tree-Based Channel Assignment (TMCP)* [8]. These methods consider the channel assignment problem at different levels: the link level, node level, or cluster level. We show that for aggregated convergecast, TMCP performs better than JFTSS and RBCA on minimum-hop routing trees, while performs worse on degree-constrained trees. For raw-data convergecast, RBCA and JFTSS perform better than TMCP, since the latter suffers from interference inside the branches due to concurrent transmissions on the same channel.
- **Impact of Routing Trees**: We investigate the effect of network topology on the schedule length, and show that for aggregated convergecast the performance can be improved by up to 10 times on degree-constrained trees using multiple frequencies as compared to that on minimum-hop trees using a single frequency. For raw-data convergecast, multi-channel scheduling on capacitated minimal spanning trees can reduce the schedule length by 50%.
- **Impact of Channel Models and Interference**: Under the setting of multiple frequencies, one simplifying assumption often made is that the frequencies are orthogonal to each other. We evaluate this assumption and show that the schedules generated may not always eliminate interference, thus causing considerable packet losses. We also evaluate and compare the two most commonly used interference models: (i) the graph-based *protocol model*, and (ii) the SINR (Signal-to-Interference-plus-Noise Ratio) based *physical model*.

The rest of the paper is organized as follows: In Section 2, we discuss related works. In Section 3, we describe the problem formulation and state our assump-

tions. In Section 4, we analyze the lower bounds on the schedule length for aggregated and raw convergecast, and propose algorithms that achieve the corresponding bounds. In Section 5, we focus on power control and multi-channel scheduling as mechanisms to eliminate interference. Section 6 explains the impact of routing topologies, and Section 7 presents detailed evaluation results. Finally, we draw our conclusions in Section 8.

2 Related Work

Fast data collection with the goal to minimize the schedule length for aggregated convergecast has been studied by us in [7], [9], and also by others in [5], [10], [11]. In [7], we experimentally investigated the impact of transmission power control and multiple frequency channels on the schedule length, while the theoretical aspects were discussed in [9], where we proposed constant factor and logarithmic approximation algorithms on geometric networks (disk graphs). Raw-data convergecast has been studied in [1], [12]–[14], where a distributed time slot assignment scheme is proposed by Gandham *et al.* [1] to minimize the TDMA schedule length for a single channel. The problem of joint scheduling and transmission power control is studied by Moscibroda [5] for constant and uniform traffic demands. Our present work is different from the above in that we evaluate transmission power control under realistic settings and compute lower bounds on the schedule length for tree networks with algorithms to achieve these bounds. We also compare the efficiency of different channel assignment methods and interference models, and propose schemes for constructing specific routing tree topologies that enhance the data collection rate for both aggregated and raw-data convergecast.

The use of orthogonal codes to eliminate interference has been studied by Annamalai *et al.* [10], where nodes are assigned time slots from the bottom of the tree to the top such that a parent node does not transmit before it receives all the packets from its children. This problem and the one addressed by Chen *et al.* [11] are for one-shot raw-data convergecast. In this work, since we construct degree-constrained routing topologies to enhance the data collection rate, it may not always lead to schedules that have low latency, because the number of hops in a tree goes up as its degree goes down. Therefore, if minimizing latency is also a requirement, then further optimization, such as constructing bounded-degree, bounded-diameter trees, is needed. A study along this line with the objective to minimize the maximum latency is presented by Pan and Tseng [15], where they assign a beacon period to each node in a Zigbee network during which it can receive data from all its children.

For raw-data convergecast, Song *et al.* [12] presented a time-optimal, energy-efficient, packet scheduling algorithm with periodic traffic from all the nodes to the sink. Once interference is eliminated, their algorithm achieves the bound that we present here, however, they briefly mention a 3-coloring channel assignment scheme, and it is not clear whether the channels are frequencies, codes, or any other method to eliminate interference. Moreover, they assume a simple interference model where each node has a circular transmission range and cumulative interference from concurrent multiple senders is avoided. Different from their work, we consider multiple *frequencies* and evaluate the performance of three different channel assignment methods together with evaluating the effects of transmission power control using realistic interference and channel models, i.e., physical interference model and overlapping channels and considering the impact of routing topologies. Song *et al.* [12] extended their work and proposed a TDMA based MAC protocol for high data rate WSNs in [16]. TreeMAC considers the differences in load at different levels of a routing tree and assigns time slots according to the depth, i.e. the hop count, of the nodes on the routing tree, such that nodes closer to the sink are assigned more slots than their children in order to mitigate congestion. However, TreeMAC operates on a single channel and achieves $1/3$ of the maximum throughput similar to the bounds presented by Gandham *et al.* [1] since the sink can receive every 3 time slots.

The problem of minimizing the schedule length for raw-data convergecast on single channel is shown to be NP-complete on general graphs by Choi *et al.* [13]. Maximizing the throughput of convergecast by finding a shortest-length, conflict-free schedule is studied by Lai *et al.* [14], where a greedy graph coloring strategy assigns time slots to the senders and prevent interference. They also discussed the impact of routing trees on the schedule length and proposed a routing scheme called *disjoint strips* to transmit data over different shortest paths. However, since the sink remains as the bottleneck, sending data over different paths does not reduce the schedule length. As we will show in this paper, the improvement due to the routing structure comes from using capacitated minimal spanning trees for raw-data convergecast, where the number of nodes in a subtree is no more than half the total number of nodes in the remaining subtrees.

The use of multiple frequencies has been studied extensively in both cellular and ad hoc networks, however, in the domain of WSN, there exist a few studies that utilize multiple channels [8], [17], [18]. To this end, we evaluate the efficiency of three particular schemes that treat the channel assignment at different levels.

3 Modeling and Problem Formulation

We model the multi-hop WSN as a graph $G = (V, E)$, where V is the set of nodes, $E = \{(i, j) \mid i, j \in V\}$ is the set of edges representing the wireless links. A designated node $s \in V$ denotes the sink. The Euclidean distance between two nodes i and j is denoted by d_{ij}. All the nodes except s are sources, which generate packets and transmit them over a routing tree to s. We denote the

spanning tree on G rooted at s by $T = (V, E_T)$, where $E_T \subseteq E$ represents the tree edges. Each node is assumed to be equipped with a single half-duplex transceiver, which prevents it from sending and receiving packets simultaneously. We consider a TDMA protocol where time is divided into slots, and consecutive slots are grouped into equal sized non-overlapping frames.

We use two types of interference models for our evaluation: the graph-based protocol model and the SINR-based physical model. In the protocol model, we assume that the interference range of a node is equal to its transmission range, i.e., two links cannot be scheduled simultaneously if the receiver of at least one link is within the range of the transmitter of the other link. In the physical model, the successful reception of a packet from i to j depends on the ratio between the received signal strength at j and the cumulative interference caused by all other concurrently transmitting nodes and the ambient noise level. Thus, a packet is received successfully at j if the signal-to-interference-plus-noise ratio, $SINR_{ij}$, is greater than a certain threshold β, i.e.,

$$SINR_{ij} = \frac{P_i \cdot g_{ij}}{\sum_{k \neq i} P_k \cdot g_{kj} + \mathcal{N}} \quad (1)$$

where P_i is the transmitted signal power at node i, $\mathcal{N}$ is the ambient noise level, and g_{ij} is the propagation attenuation (link gain) between i and j. We use a simple distance dependent path-loss model to calculate the link gains as $g_{ij} = d_{ij}^{-\alpha}$, where the path-loss exponent α is a constant between 2 and 6, whose exact value depends on external conditions of the medium (humidity, obstacles, etc.), as well as the sender-receiver distance. We assume that the level of interference is static and does not change over time. For simplicity and ease of illustration, we use the protocol model in all the figures.

We study aggregated convergecast in the context of periodic data collection where each source node generates a packet at the beginning of every frame, and raw-data convegecast for one-shot data collection where each node has only one packet to send. We assume that the size of each packet is constant. Our goal is to deliver these packets to the sink over the routing tree as fast as possible. More specifically, we aim to schedule the edges E_T of T using a minimum number of time slots while respecting the following two constraints:

- *Adjacency Constraint*: Two edges $(i, j) \in E_T$ and $(k, l) \in E_T$ cannot be scheduled in the same time slot if they are adjacent to each other, i.e., if $\{i, j\} \bigcap \{k, l\} \neq \phi$. This constraint is due to the half-duplex transceiver on each node which prevents it from simultaneous transmission and reception.
- *Interfering Constraint*: The interfering constraint depends on the choice of the interference model. In the protocol model, two edges $(i, j) \in E_T$ and $(k, l) \in E_T$ cannot be scheduled simultaneously if they are at two hop distance of each other. In the physical model, an edge $(i, j) \in E_T$ cannot be scheduled if the SINR at receiver j is not greater than the threshold β.

Since we consider data collection to be periodic in aggregated convergecast, each of the edges in E_T is scheduled only once within each frame, and this schedule is repeated over multiple frames. Thus, a *pipeline* is established after a certain frame, and then onwards the sink continues to receive aggregated packets from all the source nodes once per frame. We explain further details about the pipelining in the next section. On the other hand, in one-shot data collection for raw-data convergecast, the edges in E_T may be scheduled multiple times and no pipelining takes place. We use the terms link scheduling and node scheduling interchangeably as they are equivalent in our case. Note that the two other scenarios, which we do not consider in this paper due to space constraints, are one-shot aggregated convergecast and periodic raw-data convergecast.

The key difference in terms of scheduling between periodic and one-shot data collection is that a node in the periodic case does not have to wait for data from its children before being scheduled. This is because a link is scheduled only once within each frame and each node generates a packet in the beginning of every frame, so a pipelining is eventually established. However, in the case of one-shot data collection, a node needs to wait for data from its children before being scheduled, which we refer to as the *causality constraint*.

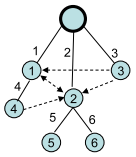
To summarize the steps in our design, we start with tree construction and then continue with interference-aware scheduling. If the nodes can control their transmission power, scheduling phase is coupled with a transmission power control algorithm. If the nodes can change their operating frequency, channel scheduling can be coupled with time slot scheduling as it is the case with the JFTSS algorithm (Section 5.2.1) or first channels are assigned and then time slot scheduling continues as in the case of RBCA explained in Section 5.2.3. However, the TMCP algorithm (Section 5.2.2), considers tree construction and channel assignment jointly and then does the scheduling of time slots.

4 TDMA Scheduling of Convergecasts

In this section, we first focus on periodic aggregated convergecast and then on one-shot raw-data convergecast. Our objective is to calculate the minimum achievable schedule lengths using an interference-aware TDMA protocol. We first consider the case where the nodes communicate on the same channel using a constant transmission power, and then discuss improvements using transmission power control and multiple frequencies in the next section.

4.1 Periodic Aggregated Convergecast

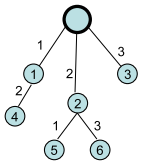
In this section, we consider the scheduling problem where packets are aggregated. Data aggregation is a



(a)

	Frame 1						Frame 2					
	S1	S2	S3	S4	S5	S6	S1	S2	S3	S4	S5	S6
s	1	2	3	-	-	-	{1,4}	{2,5,6}	3	-	-	-
1	-	-	-	4	-	-	-	-	-	4	-	-
2	-	-	-	-	5	6	-	-	-	-	5	6

(b)



(c)

Fig. 1: Aggregated convergecast and pipelining: (a) Schedule length of 6 in the presence of interfering links. (b) Node ids from which (aggregated) packets are received by their corresponding parents in each time slot over different frames. (c) Schedule length of 3 using BFS-TIMESLOTASSIGNMENT when all the interfering links are eliminated.

commonly used technique in WSN that can eliminate redundancy and minimize the number of transmissions, thus saving energy and improving network lifetime [19]. Aggregation can be performed in many ways, such as by suppressing duplicate messages; using data compression and packet merging techniques; or taking advantage of the correlation in the sensor readings.

We consider continuous monitoring applications where perfect aggregation is possible, i.e., each node is capable of aggregating all the packets received from its children as well as that generated by itself into a single packet before transmitting to its parent. The size of aggregated data transmitted by each node is constant and does not depend on the size of the raw sensor readings. Typical examples of such aggregation functions are MIN, MAX, MEDIAN, COUNT, AVERAGE, etc.

In Fig. 1(a) and 1(b), we illustrate the notion of pipelining in aggregated convergecast and that of a schedule length on a network of 6 source nodes. The solid lines represent tree edges, and the dotted lines represent interfering links. The numbers beside the links represent the time slots at which the links are scheduled to transmit, and the numbers inside the circles denote node id's. The entries in the table 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 does not have packets from nodes 5 and 6; however, as the schedule is repeated, it receives aggregated packets from 2, 5, and 6 in slot 2 of the next frame. Similarly, the sink also receives aggregated packets from nodes 1 and 4 starting from slot 1 of frame 2. The entries $\{1, 4\}$ and $\{2, 5, 6\}$ in the table represent single packets comprising aggregated data from nodes 1 and 4, and from nodes 2, 5, and 6, respectively. Thus, a pipeline is established from frame 2, and the sink continues to receive aggregated packets from *all* the nodes once every 6 time slots. Thus, the minimum schedule length is 6.

4.1.1 Lower Bound on Schedule Length

We first consider aggregated convergecast when all the interfering links are eliminated by using transmission power control or multiple frequencies. Although the problem of minimizing the schedule length is NP-complete on general graphs, we show in the following that once interference is eliminated, the problem reduces to one on a tree, and can be solved in polynomial time. To this end, we first give a lower bound on the schedule length, and then propose a time slot assignment scheme that achieves the bound.

LEMMA 1: *If all the interfering links are eliminated, the schedule length for aggregated convergecast is lower bounded by $\Delta(T)$, where $\Delta(T)$ is the maximum node degree in the routing tree T.*

Proof: If all the interfering links are eliminated, the scheduling problem reduces to one on a tree. Now since each of the tree edges needs to be scheduled only once within each frame, it is equivalent to edge coloring on a graph, which needs number of colors at least equal to the maximum node degree. □

Once all the interfering links are eliminated, concurrency is still limited by the adjacency constraint due to the half-duplex transceivers, which prevents a parent from transmitting when it is already receiving from its children, or when its parent is transmitting.

4.1.2 Assignment of Timeslots

Given the lower bound $\Delta(T)$ on the schedule length in the absence of interfering links, we now present a time slot assignment scheme in Algorithm 1, called BFS-TIMESLOTASSIGNMENT, that achieves this bound.

In each iteration of BFS-TIMESLOTASSIGNMENT (lines 2-6), an edge e is chosen in the Breadth First Search (BFS) order starting from any node, and is assigned the minimum time slot that is different from all its adjacent edges respecting interfering constraints. Note that, since we evaluate the performance of this algorithm also for the case when the interfering links are present, we check for the corresponding constraint in line 4; however, when interference is eliminated this check is redundant. The algorithm runs in $O(|E_T|^2)$ time and minimizes the schedule length when there are no interfering links, as proved in Theorem 1. To illustrate, we show the same network of Fig. 1(a) in 1(c) with all the interfering links removed, and so the network is scheduled in 3 time slots.

Algorithm 1 BFS-TIMESLOTASSIGNMENT

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 minimum time slot t to edge e respecting adjacency and interfering constraints
5. $E_T \leftarrow E_T \setminus \{e\}$
6. **end while**

Although BFS-TIMESLOTASSIGNMENT may not be an approximation to ideal scheduling under the physical interference model, it is a heuristic that can achieve the

lower bound if all the interfering links are eliminated. Therefore, together with a method to eliminate interference the algorithm can optimally schedule the network.

THEOREM 1: *If all the interfering links are eliminated, the schedule length for aggregated convergecast achieved by* BFS-TIMESLOTASSIGNMENT *is the minimum, i.e.,* $\Delta(T)$.

Proof: The proof is by induction on i. Let $T^i = (V^i, E^i_T)$ denote the subtree of T in the i^{th} iteration constructed in the BFS order, where E^i_T comprises all the edges that are assigned a slot, and V^i comprises the set of nodes on which the edges in E^i_T are incident. Note that, $|E^i_T| = i$, because at every iteration exactly one edge is assigned a slot. For $i = 1$, clearly the number of slots used is 1, equal to $\Delta(T^1)$.

Now, assume that the number of slots $N(i)$ needed to schedule the edges in T^i is $\Delta(T^i)$. In the $(i+1)^{th}$ iteration, after assigning a slot to the next edge in BFS order, the number of slots needed in T^{i+1} can either remain the same as before, or increase by one. Thus,

$$N(i+1) = \max\{N(i), N(i)+1\} \quad (2)$$

If it remains the same, $N(i+1)$ is still the maximum degree of T^{i+1} at end of $(i+1)^{th}$ iteration. Otherwise, if it increases by one, the new edge must be incident on a node v^*, common to both T^i and T^{i+1}, such that the number of incident edges on v^* that were already assigned a time slot at the end of i^{th} iteration was $\Delta(T^i)$. This is so because in the BFS traversal, *all* the edges incident on a node are assigned a slot first before moving on to the next node, and because the slot assigned to the new edge is the minimum possible that is different from all that already assigned to the edges incident on v^* until the i^{th} iteration. Thus, at the end of $(i+1)^{th}$ iteration, the number of slots used $N(i)+1$ is equal to the number of assigned edges incident on v^* which, in turn, equals $\Delta(T^{i+1})$. This proves the inductive step. Therefore, it holds at every iteration of the algorithm until the end when $i = |V| - 2$, yielding a schedule length equal to the maximum degree $\Delta(T) = \Delta(T^{|V|-1})$. Now, since assigning different time slots to the adjacent edges of T is equivalent to edge coloring T, which requires at least $\Delta(T)$ colors, the schedule length is minimum. □

4.2 One-Shot Raw-Data Convergecast

In this section, we consider one-shot data collection where every sensor reading is equally important, and so aggregation may not be desirable or even possible. Thus, each of the packets has to be individually scheduled at each hop *en route* to the sink. As before, we focus on minimizing the schedule length. Unlike in the case of periodic aggregated convergecast where a pipelining takes place and each of the tree edges is scheduled only once within each frame, here the edges could be scheduled multiple times and there is no pipelining.

The problem of minimizing the scheduling length for raw-data convergecast is proved to be NP-complete even under the protocol interference model by a reduction from the well known *Partition Problem* [13]. Before getting into the details, we first define the following terms: a *branch* is defined as a subtree containing the sink as an endpoint; a *top-subtree* is defined as a subtree that has a child of the sink as its root. For instance, in Fig. 3, the branches are $\{s, 1, 4\}$, $\{s, 2, 5, 6\}$, and $\{s, 3, 7\}$, while the top-subtrees are $\{1, 4\}$, $\{2, 5, 6\}$, and $\{3, 7\}$.

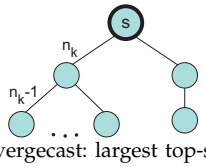


Fig. 2: Raw-data convergecast: largest top-subtree with n_k nodes.

4.2.1 Lower Bound on Schedule Length

As mentioned in Section 4.1.1, if all the interfering links are eliminated using multiple frequencies, the only limiting factor in minimizing the schedule length is the half-duplex transceivers. In the following, we give a lower bound on the schedule length under this scenario.

LEMMA 2: *If all the interfering links are eliminated, the schedule length for one-shot raw-data convergecast is lower bounded by* $\max(2n_k - 1, N)$, *where* n_k *is the maximum number of nodes in any top-subtree of the routing tree, and* N *is the number of sources in the network.*

Proof: Let n_i denote the number of nodes in top-subtree i. Order the top-subtrees in non-increasing order of their sizes: $n_k \geq n_{k-1} \geq \ldots \geq n_1$. Consider the routing tree shown in Fig. 2. Since the nodes cannot receive multiple packets simultaneously, N is a trivial lower bound to receive all the packets. Next, consider the largest top-subtree k, the root of which has to transmit n_k packets to the sink, and the children of this root have to forward $n_k - 1$ packets in total. Because of the half-duplex transceivers, time slots assigned to the root of this top-subtree must be distinct from all those assigned to its children. Thus, in total we need at least $n_k + (n_k - 1) = 2n_k - 1$ distinct time slots. □

We note that this bound of $\max(2n_k - 1, N)$, which applies only when all the interfering links are removed, is smaller than the lower bound of $3N$ for general networks and that of $\max(3n_k - 3, N)$ for tree networks, as computed by Gandham *et al.* [1] for the 2-hop interference model. They proposed a time slot assignment scheme for tree networks, which requires each node to maintain a buffer that stores at most two packets and minimizes the schedule length. In the following, we describe a time slot assignment scheme that computes a schedule of length exactly equal to the lower bound when interference is eliminated and does not require to store more than one packet in buffers at any time.

4.2.2 Assignment of Timeslots

We now describe a time slot assignment scheme in Algorithm 2, called LOCAL-TIMESLOTASSIGNMENT, which is run *locally* by each node at every time slot. The key idea is to: (i) schedule transmissions in parallel along multiple branches of the tree, and (ii) keep the sink busy in receiving packets for as many time slots as possible.

Because the sink can receive from the root of at most one top-subtree in any time slot, we need to decide which top-subtree should be made active. We assume that the sink is aware of the number of nodes in each top-subtree. Each source node maintains a buffer and its associated state, which can be either *full* or *empty* depending on whether it contains a packet or not. Our algorithm does not require any of the nodes to store more than one packet in their buffer at any time. We initialize all the buffers as full, and assume that the sink's buffer is always full for the ease of explanation.

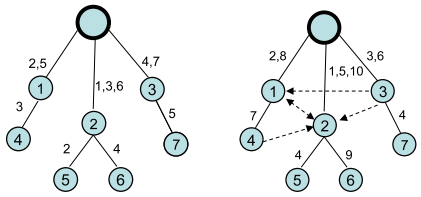
Algorithm 2 LOCAL-TIMESLOTASSIGNMENT

1. *node*.buffer = *full*
2. **if** {*node* is sink} **then**
3. Among the eligible top-subtrees, choose the one with the largest number of total (remaining) packets, say top-subtree i
4. Schedule link $(root(i), s)$ respecting interfering constraint
5. **else**
6. **if** {*node*.buffer == *empty*} **then**
7. Choose a random child c of *node* whose buffer is *full*
8. Schedule link $(c, node)$ respecting interfering constraint
9. c.buffer = *empty*
10. *node*.buffer = *full*
11. **end if**
12. **end if**

The first block of the algorithm in lines 2-4 gives the scheduling rules between the sink and the roots of the top-subtrees. We define a top-subtree to be *eligible* if its root has at least one packet to transmit. For a given time slot, we schedule the root of an eligible top-subtree which has the *largest* number of total (remaining) packets. If none of the top-subtrees are eligible, the sink does not receive any packet during that time slot.

Inside each top-subtree, nodes are scheduled according to the rules in lines 5-12. We define a subtree to be *active* if there are still packets left in the subtree (excluding its root) to be relayed. If a node's buffer is empty and the subtree rooted at this node is active, we schedule one of its children at random whose buffer is not empty. Our algorithm guarantees (as proved in Lemma 3) that in an active subtree there will always be at least one child whose buffer is not empty, and so whenever a node empties its buffer, it will receive a packet in the next time slot, thus emptying buffers from the bottom of the subtree to the top.

We run through an example shown in Fig. 3(a) to explain the algorithm. In the first time slot, since the eligible top-subtree containing the largest number of remaining packets is $\{2, 5, 6\}$, we schedule the link $(2, s)$, and the sink receives a packet from node 2 in slot 1. In the second time slot, the eligible top-subtrees are $\{1, 4\}$ and $\{3, 7\}$, both of which have 2 remaining packets. We choose one of them at random, say $\{1, 4\}$, and schedule the link $(1, s)$. Also, in the same time slot since node 2's buffer is empty, it chooses one of its children at random, say node 5, and schedule the link $(5, 2)$. In the third time slot, the eligible top-subtrees are $\{2, 5, 6\}$ and $\{3, 7\}$, both of which have 2 remaining packets. We choose the first one at random and schedule the link



(a) (b)

Fig. 3: Raw-data convergecast using algorithm LOCAL-TIMESLOTASSIGNMENT: (a) Schedule length of 7 when all the interfering links are removed. (b) Schedule length of 10 when the interfering links are present.

$(2, s)$, and so the sink receives node 5's packet (relayed by node 2). We also schedule the link $(4, 1)$ in the third time slot because node 1's buffer is empty at this point. This process continues until all the packets are delivered to the sink, yielding an assignment that requires 7 time slots. Note that, in this example, $2n_k - 1 = 5$, and so $\max(2n_k - 1, N) = 7$. In Fig. 3(b), we show an assignment when all the interfering links are present, yielding a schedule length of 10.

In the following, we prove that the algorithm requires exactly $\max(2n_k - 1, N)$ slots when all the interfering links are eliminated. Before giving the details of the proof, we first highlight the two key insights of the algorithm: (i) the sink is kept busy in receiving packets for as many time slots as possible, and (ii) a node's buffer is not empty for 2 or more consecutive time slots so long as the subtree rooted at this node is active. The first one is evident from the scheduling rule between the sink and the top-subtrees. We prove the second insight in the following lemma.

LEMMA 3: *In an active subtree, a node with an empty buffer always has a child and a parent whose buffers are full.*

Proof: We prove it by induction on time slot t. The parent and grandparent of node i are denoted by $p(i)$ and $gp(i)$; similarly a child and a grandchild of i are denoted by $c(i)$ and $gc(i)$, respectively. Slightly abusing notation, we also use these symbols to denote the state of the buffers on the respective nodes.

At $t = 1$, the lemma is trivially true because all the buffers are full. Suppose the lemma holds for $t = k$, i.e., every node whose buffer is empty has a child and a parent whose buffers are full. At $t = k + 1$, each node with an empty buffer schedules one of its children whose buffer is full. The following two situations can occur:

- Node i is full, while $p(i)$ and $c(i)$ are both empty.
- Nodes i and $p(i)$ are both full, while $c(i)$ is empty.

For the first case, we need to show that both $p(i)$ and $c(i)$ (since now they are empty) have a child and a parent whose buffers are full. Clearly, $p(i)$ has a child with a full buffer because i is now full. Similarly, $p(i)$ also has a parent with a full buffer because a transmission took place from $p(i)$ to its parent at $t = k + 1$. For the latter, $c(i)$ has a parent with a full buffer because transmission took place from $c(i)$ to i at $t = k + 1$. If the child of $c(i)$, i.e., $gc(i)$, was empty at $t = k$, then $gc(i)$ also had a child with a full buffer because the lemma was true at $t = k$. Therefore, at $t = k + 1$ the child of $gc(i)$ transmits and fills up its parent's buffer. Otherwise, if $gc(i)$ was full at

$t = k$, then it also remains full at $t = k + 1$ because it cannot transmit to its parent $c(i)$, which was full at t.

For the second case, $c(i)$ transmitted and $p(i)$ did not. For this to happen, $gp(i)$ was full at $t = k$ and either empties or remains full at $t = k + 1$. If it empties, $gp(i)$ has a parent with a full buffer because it transmitted at $t = k+1$, and also has a child with a full buffer because $p(i)$ did not transmit. If it remains full, at $t = k+1$ nodes i, $p(i)$, and $gp(i)$ are full, $c(i)$ is empty and $gc(i)$ is full as we showed in the first case. So, the lemma holds for $t = k + 1$, and the proof follows. □

THEOREM 2: *If all the interfering links are eliminated, the schedule length for raw-data convergecast achieved by algorithm* LOCAL-TIMESLOTASSIGNMENT *is the minimum, i.e.,* $\max(2n_k - 1, N)$.

Proof: Let n_i be the number of nodes in top-subtree i. Order the top-subtrees in non-increasing order of their sizes: $n_k \geq n_{k-1} \geq \ldots \geq n_1$. Suppose $n_k > \sum_{i=1}^{k-1} n_i$; then $\max(2n_k-1, N) = 2n_k-1$. From Lemma 1, we know that it takes at least $2n_k - 1$ slots to schedule all the packets originated in top-subtree k. Out of these, the sink can use at most n_k-1 slots to receive packets from the other top-subtrees, which have a total of at most $n_k - 1$ packets. Also, when $n_k > \sum_{i=1}^{k-1} n_i$, the root of the largest top-subtree k gets scheduled once in every two time slots. Therefore, the schedule length is at most $2n_k - 1$.

Now suppose $n_k \leq \sum_{i=1}^{k-1} n_i$; then $\max(2n_k - 1, N) = N$. We need to show that there always exists an eligible top-subtree to complement for the largest one when it is not eligible. In this case, the sink will receive packets in every slot, because otherwise it remains idle during some time slots and the first condition $n_k > \sum_{i=1}^{k-1} n_i$ will be met. Thus, we will prove that the algorithm keeps the inequality $n_k \leq \sum_{i=1}^{k-1} n_i$ as an invariant.

In any given time slot t, the algorithm schedules an eligible top-subtree that has the largest number of remaining packets. At slot $t + 1$, therefore, we have $n_k = n_k - 1$, and the following three cases might arise:

- Top-subtree k still has the largest number of remaining packets with $n_k \geq n_{k-1} \geq \ldots \geq n_1$. Then, the root of k is again chosen to transmit at $t + 1$, and the inequality still holds as $n_k - 1 \leq \sum_{i=1}^{k-1} n_i$.
- Top-subtree k and at least another one, say j, have an equal number of remaining packets. Then, the root of j is chosen, and the inequality still holds because $n_j - 1 \leq \sum_{i=1}^{k-1} n_i - 1$ (since $n_j = n_k - 1$).
- Top-subtree k does not have the largest number of remaining packets, implying that there were other top-subtrees with an equal number of packets left as k in slot t. Then, the root of a new largest top-subtree j is chosen, and the inequality holds since $n_j - 1 \leq \sum_{i=1}^{k-1} n_i - 1$ (since $n_j = n_k$).

Thus, the algorithm keeps the inequality as an invariant, and there always exists a top-subtree that can be alternately scheduled with the largest top-subtree. When $n_k = 1$, $\sum_{i=1}^{k-1} n_i - 1 = 1$, which means that there are 2 packets left at two different top-subtrees that can be scheduled in alternate slots. Since this inequality holds for all the N steps, the sink always finds a top-subtree to receive packets from, and therefore it takes N slots. Moreover, Lemma 1 implies that a top-subtree becomes eligible after a transmission because its root is filled up in the next slot. Therefore, the theorem follows. □

5 IMPACT OF INTERFERENCE

So far, we have focused on computing spatial-reuse TDMA schedules where transmissions take place on the same frequency at a constant transmission power. In this section, we focus on different methods to mitigate the effects of interference on the schedule length. First, we discuss the benefits of using transmission power control and explain the basics of a possible algorithm. Then we discuss the advantages of using multiple channels by considering 3 different channel assignment schemes.

5.1 Transmission Power Control

In wireless networks, excessive interference can be eliminated by using transmission power control [6], [20], i.e., by transmitting signals with just enough power instead of maximum power. To this end, we evaluate the impact of transmission power control on fast data collection using discrete power levels, as opposed to a continuous range where an unbounded improvement in the asymptotic capacity can be achieved by using a non-linear power assignment [5]. We first explain the basics of one particular algorithm that we use in our evaluations in Section 7.

The algorithm proposed by El Batt *et al.* [6] is a cross layer method for joint scheduling and power control and it is an optimal distributed algorithm to improve the throughput capacity of wireless networks. The goal is to find a TDMA schedule that can support as many transmissions as possible in every time slot. It has two phases: (i) scheduling and (ii) power control that are executed at every time slot. First the scheduling phase searches for a *valid transmission schedule*, i.e., largest subset of nodes, where no node is to transmit and receive simultaneously, or to receive from multiple nodes simultaneously. Then, in the given valid schedule the power control phase iteratively searches for an *admissible schedule* with power levels chosen to satisfy all the interfering constraints. In each iteration, the scheduler adjusts the power levels depending on the current RSSI at the receiver and the SINR threshold according to the iterative rule: $P_{new} = \frac{\beta}{SINR} \cdot P_{current}$. According to this rule, if a node transmits with a power level higher than what is required by the threshold value, it should decrease its power and if it is below the threshold it should increase its transmission power, within the available range of power levels on the radio. If all the nodes meet the interfering constraint, the algorithm proceeds with the schedule calculation for the next time slot. On the other hand, if the maximum number of iterations is reached and there are nodes

which cannot meet the interfering constraint, the algorithm excludes the link with minimum SINR from the schedule and restarts the iterations with the new subset of nodes. The power control phase is repeated until an admissible transmission scenario is found.

5.2 Multi-Channel Scheduling

Multi-channel communication is an efficient method to eliminate interference by enabling concurrent transmissions over different frequencies [21]. Although typical WSN radios operate on a limited bandwidth, their operating frequencies can be adjusted, thus allowing more concurrent transmissions and faster data delivery. Here, we consider fixed-bandwidth channels, which are typical of WSN radios, as opposed to the possibility of improving link bandwidth by consolidating frequencies. In this section, we explain three channel assignment methods that consider the problem at different levels allowing us to study their pros and cons for both types of convergecast. These methods consider the channel assignment problem at different levels: the link level (JFTSS), node level (RBCA), or cluster level (TMCP).

5.2.1 Joint Frequency Time Slot Scheduling (JFTSS)

JFTSS offers a greedy joint solution for constructing a maximal schedule, such that a schedule is said to be *maximal* if it meets the adjacency and interfering constraints, and no more links can be scheduled for concurrent transmissions on any time slot and channel without violating the constraints. Approximation bounds on JFTSS for single-channel systems and its comparison with multi-channel systems are discussed in [22] and [23], respectively.

JFTSS schedules a network starting from the link that has the highest number of packets (load) to be transmitted. When the link loads are equal, such as in aggregated convergecast, the most constrained link is considered first, i.e., the link for which the number of other links violating the interfering and adjacency constraints when scheduled simultaneously is the maximum. The algorithm starts with an empty schedule and first sorts the links according to the loads or constraints. The most loaded or constrained link in the first available slot-channel pair is scheduled first and added to the schedule. All the links that have an adjacency constraint with the scheduled link are excluded from the list of the links to be scheduled at a given slot. The links that do not have an interfering constraint with the scheduled link can be scheduled in the same slot and channel whereas the links that have an interfering constraint should be scheduled on different channels, if possible. The algorithm continues to schedule the links according to the most loaded (or most constrained) metric. When no more links can be scheduled for a given slot, the scheduler continues with scheduling in the next slot. Fig. 4(a) shows the same tree given in Fig. 1(a) which is scheduled according to JFTSS where aggregated data

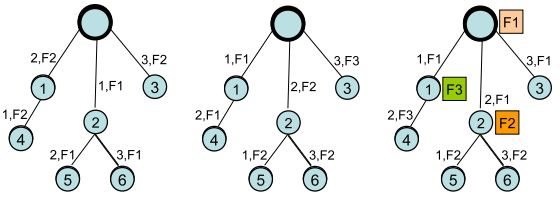


Fig. 4: Scheduling with multi-channels for aggregated convergecast: (a) Schedule generated with JFTSS. (b) Schedule generated with TMCP. (c) Schedule generated with RBCA. (b) Schedule generated with RBCA.

is collected. JFTSS starts with link $(2, sink)$ on frequency 1 and then schedules link $(4, 1)$ next on the first slot on frequency 2. Then, links $(5, 2)$ on frequency 1 and $(1, sink)$ on frequency 2 are scheduled on the second slot and links $(6, 2)$ on frequency 1 and $(3, sink)$ on frequency 2 are scheduled on the last slot.

An advantage of JFTSS is that it is easy to incorporate the physical interference model, however, it is hard to have a distributed solution since the interference relationship between all the links must be known.

5.2.2 Tree-Based Multi-Channel Protocol (TMCP)

TMCP is a greedy, tree-based, multi-channel protocol for data collection applications [8]. It partitions the network into multiple subtrees and minimizes the *intra-tree* interference by assigning different channels to the nodes residing on different branches starting from the top to the bottom of the tree. Figure 4(b) shows the same tree given in Fig. 1(a) which is scheduled according to TMCP for aggregated data collection. Here, the nodes on the leftmost branch is assigned frequency $F1$, second branch is assigned frequency $F2$ and the last branch is assigned frequency $F3$ and after the channel assignments, time slots are assigned to the nodes with the BFS-TimeSlotAssignment algorithm. The advantage of TMCP is that it is designed to support convergecast traffic and does not require channel switching. However, contention inside the branches is not resolved since all the nodes on the same branch communicate on the same channel.

5.2.3 Receiver-Based Channel Assignment (RBCA)

In our previous work [7], we proposed a channel assignment method called RBCA where we statically assigned the channels to the receivers (parents) so as to remove as many interfering links as possible. In RBCA, the children of a common parent transmit on the same channel. Every node in the tree, therefore, operates on at most two channels, thus avoiding pair-wise, per-packet, channel negotiation overheads. The algorithm initially assigns the same channel to all the receivers. Then, for each receiver, it creates a set of interfering parents based on SINR thresholds and iteratively assigns the next available channel starting from the most interfered parent (the parent with the highest number of interfering links). However, due to adjacent channel overlaps, SINR values at the receivers may not always be high enough to tolerate interference, in which case the channels are assigned according to the ability of the transceivers to

reject interference. We proved approximation factors for RBCA when used with greedy scheduling in [9]. Figure 4(c) shows the same tree given in Fig. 1(a) scheduled with RBCA for aggregated convergecast. Initially all nodes are on frequency $F1$. RBCA starts with the most interfered parent, node 2 in this example, and assigns $F2$. Then it continues to assign $F3$ to node 3 as the second most interfered parent. Since all interfering parents are assigned different frequencies sink can receive on $F1$.

6 Impact of Routing Trees

Besides transmission power control and multiple channels, the network topology and the degree of connectivity also affect the scheduling performance. In this section, we describe schemes to construct topologies with specific properties that help to reduce the schedule length.

6.1 Aggregated Data Collection

We first construct balanced trees and compare their performance with unbalanced trees. We observe that in both cases the sink often creates a high-degree bottleneck. To overcome this, we then propose a heuristic, as described in Algorithm 3, by modifying Dijkstra's shortest path algorithmto construct *degree-constrained* treesNote that constructing such a degree-constrained tree is NP-hard. Each source node i in our heuristic keeps track of the number of its children, $C(i)$, which is initialized to 0, and a hop count to the sink, $HC(i)$, which is initialized to ∞. The algorithm starts with the sink node, and adds a node $i' \notin T$ at every iteration to the tree such that $HC(i')$ is minimized. It stops when $|T| = |V|$, or when no more nodes can be added to the tree because the neighbors of all these new nodes have reached the limit on their maximum degree. Consequently, in this latter situation, the heuristic might not always generate a spanning tree. In our evaluation presented in Section 7.3, we consider only those instances of the topologies where spanning trees with the specified degree constraint are produced.

To illustrate the gains of degree-constrained trees, consider the case when all the N nodes are in range of each other and that of the sink. If the nodes select their parents according to minimum-hop without a degree constraint, then all of them will select the sink, and this will give a schedule length of N. However, if we limit the number of children per node to 2, then this will result in two subtrees rooted at the sink, and if there are enough frequencies to eliminate interference, the network can be scheduled using only 2 time slots, thus achieving a factor of $N/2$ reduction in the schedule length.

Algorithm 3 Degree-Constrained Trees

1. **Input**: $G(V, E)$, s, max_degree
2. $T \leftarrow \{s\}$
3. **for all** $i \in V$ **do**
4. $\quad C(i) \leftarrow 0$; $HC(i) \leftarrow \infty$
5. **end for**
6. $HC(s) \leftarrow 0$
7. **while** $|T| \neq |V|$ **do**
8. $\quad$ Choose $i' \notin T$ such that:
9. $\quad\quad$ (a) $(i, i') \in E$, for some $i \in T$ with $C(i) < max\_degree - 1$
10. $\quad\quad$ (b) $HC(i')$ is minimized
11. $\quad T \leftarrow T \cup \{i'\}$
12. $\quad HC(i') = HC(i) + 1$
13. $\quad C(i) \leftarrow C(i) + 1$
14. $\quad$ **if** $\forall i \in V$, $C(i) = max\_degree$ **then**
15. $\quad\quad$ **break**
16. $\quad$ **end if**
17. **end while**

6.2 Raw Data Collection

As emphasized in [13], routing trees that allow more parallel transmissions do not necessarily result in small schedule lengths. For instance, the schedule length is N for a network connected as a star topology, whereas it is $(2N - 1)$ for a line topology once interference is eliminated. Theorem 1 suggests that the routing tree should be constructed such that all the branches have a balanced number of nodes and the constraint $n_k < (N + 1)/2$ holds. In this section, we construct such routing trees.

A balanced tree satisfying the above constraint is a variant of a *capacitated minimal spanning tree* (CMST) [24]. The CMST problem, which is known to be NP-complete, is to determine a minimum-hop spanning tree in a vertex weighted graph such that the weight of every subtree linked to the root does not exceed a prescribed capacity. In our case, the weight of each link is 1, and the prescribed capacity is $(N + 1)/2$. Here, we propose a heuristic, as described in Algorithm 4, based on the greedy scheme presented by Dai *et al.* [25], which solves a variant of the CMST problem by searching for routing trees with an equal number of nodes on each branch. We augment their scheme with a new set of rules and grow the tree hop by hop outwards from the sink. We assume that the nodes know their minimum-hop counts to sink.

Rule 1: Nodes with single potential parents are connected first.

Rule 2: For nodes with multiple potential parents, we first construct their *growth sets* (GS) and choose the one with the largest cardinality for further processing, breaking ties based on the smallest id. We define the growth set of a node as the set of neighbors (potential children) that are not yet connected to the tree and have larger hop counts.

Rule 3: Once a node is chosen based on the growth sets according to Rule 2, we construct *search sets* (SS) to decide which potential branch the node should be added to. A search set is thus branch-specific and includes the nodes that are not yet connected to the tree and are neighbors of a node that are at a higher hop count. In particular, if the chosen node has access to branch b, and has a neighbor that can connect to *only* branch b if b is selected, then this neighbor and its potential children are included in the search set for b. However, if the neighbor has access to at least one other branch even after b is selected, then it is not included in the search set.

The search sets guarantee that the choices for the nodes at longer hops to join a particular branch are

Algorithm 4 CAPACITATED-MINIMALSPANNINGTREE

1. **Input**: $G(V, E)$, s
2. **Initialize**:
3. $B \leftarrow$ roots of top subtrees // *the branches*
4. $T \leftarrow \{s\} \cup B$
5. $\forall i \in V$, $GS(i) \leftarrow$ unconnected neighbors of i at further hops
6. $\forall b \in B$, $W(b) \leftarrow 1$
7. $h \leftarrow 2$
8. **while** $h \neq max\_hop\_count$ **do**
9. $N_h \leftarrow$ unconnected nodes at hop distance h
10. Connect nodes N'_h that have a single potential parent: $T \leftarrow T \bigcup N'_h$
11. Update $N_h \leftarrow N_h \setminus N'_h$
12. Sort N_h in non-increasing order of $|GS|$
13. **for all** $i \in N_h$ **do**
14. **for all** $b \in B$ to which i can connect **do**
15. Construct $SS(i, b)$
16. **end for**
17. Connect i to b for which $W(b) + |SS(i, b)|$ is minimum
18. Update $GS(i)$ and $W(b)$
19. $T \leftarrow T \bigcup \{i\} \bigcup SS(i, b)$
20. **end for**
21. $h \leftarrow h + 1$
22. **end while**

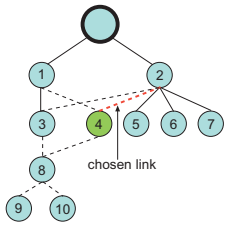


Fig. 5: Balanced tree construction: Node 4 is attached to $b2$ based on the search sets; load on both $b1$ and $b2$ is 5.

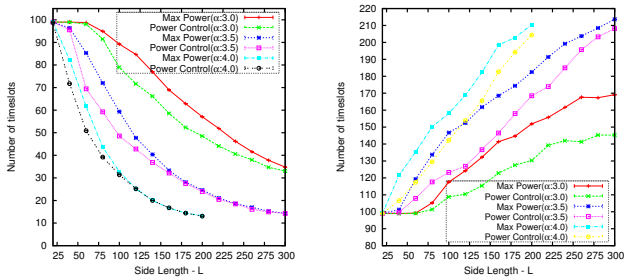
not limited by the decision of the joining node. This balances out the number of nodes on different branches and prevents one to grow faster than others. Once the search sets are constructed, we choose the branch for which the sum of its load (W) and the size of the search set is minimum.

To illustrate the merit of search sets, consider the situation shown in Fig. 5. Dotted lines represent potential communication links and solid lines represent already included tree edges. At this point, node 4 is being processed, and the loads on branches $b1$ and $b2$ are 2 and 4, respectively, where bi denote the branch rooted at node i. The search set $SS(4, b1)$ is $\{8, 9, 10\}$, because the neighbor node 8 has access to only $b1$ if $b1$ is selected by node 4. However, the search set $SS(4, b2)$ is empty, because the neighbor node 8 has access to another branch $b1$ (via node 3). Therefore, the sum of the load and the size of the search set for $b1$ is 5, and that for $b2$ is 4. So we attach node 4 to $b2$, and in the next step attach node 8 to $b1$. This balances out the number of nodes over the two branches.

7 EVALUATION

In this section, we evaluate the impact of transmission power control, multiple channels, and routing trees on the scheduling performance for both aggregated and raw-data convergecast.

We deploy nodes randomly in a region whose dimensions are varied between 20×20 m^2 and 300×300 m^2



(a) (b)

Fig. 6: Scheduling on minimum-hop trees with and without power control: (a) Aggregated convergecast. (b) Raw-data convergecast.

to simulate different levels of density. The number of nodes is kept fixed at 100. For different parameters, we average each point over 1000 runs. We use an exponential path-loss model for signal propagation with the path-loss exponent α varying between 3 and 4, which is typical for indoor environments. We also use the physical interference model and simulate the behavior of CC2420 radios that are used on Telosb and TmoteSky motes and are capable of operating on 16 different frequencies. The transmission power can be adjusted between -24 dBm and 0 dBm over 8 different levels, and the SINR threshold is set to $\beta = -3$ dB[1]. We first evaluate the schedule length for single-channel TDMA, and then its improvement using transmission power control, multiple channels, and routing trees.

7.1 Impact of Transmission Power Control

We investigate two cases: (i) when nodes transmit at maximum power, and (ii) when nodes adjust their transmission power according to the algorithm described in Section 5.1. In both cases, nodes communicate on the same channel and use minimum-hop routing trees. In the first case, time slots are assigned according to BFS-TIMESLOTASSIGNMENT for aggregated data, and according to LOCAL-TIMESLOTASSIGNMENT for raw data. In the second case, we follow the scheduling rules in [6].

7.1.1 *Aggregated Convergecast*

Fig. 6(a) shows the variation of schedule length with density for different values of α on minimum-hop trees. We observe that the schedule length decreases as the deployment gets sparser. This happens because at low densities the interference is less, and so more concurrent transmissions can take place. In the densest deployment ($L = 20$) when all the nodes are within the range of each other, the sink is the only parent, and the network is scheduled in 99 time slots regardless of power control. However, in sparser scenarios, using power control the network can be scheduled with fewer time slots as the level of interference goes down. We achieve a $10 - 20\%$ reduction in schedule length for the best case.

1. Due to variation in signal strength, a fading margin can be included such that some of the packets can still be captured if the RSSI is slightly lower than the threshold. Such a model [26] can easily be incorporated in our experiments, in which case retransmissions of lost packets should also be considered in calculating the schedule length.

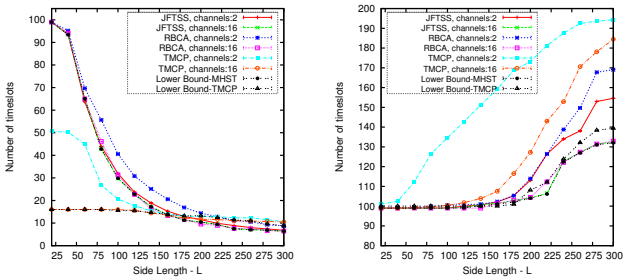


Fig. 7: Scheduling on minimum-hop trees with multiple channels: (a) Aggregated convergecast. (b) Raw-data convergecast.

We also observe that power control is more effective in reducing the schedule length for denser deployments than in sparser ones where the results tend to be similar. This is due to the discrete power levels and limited power range. Moreover, due to the -95 dBm threshold for the transceivers to be able to decode a signal successfully, further power reduction is limited.

7.1.2 Raw Data Convergecast

For raw-data convergecast, we observe in Fig. 6(b) that the schedule length increases as the network gets sparser on minimum-hop trees. This is counter intuitive because in sparse networks the reuse of slots should be higher which would reduce the schedule length. However, as the network gets sparser, the number of nodes that can directly reach the sink decreases and packets have to be relayed over more hops. Thus, more packets need to be scheduled than in a single-hop. We see that the number of packets to be scheduled increases faster than the reuse ratio. In the densest setting where all the nodes can directly reach the sink, the schedule length is 99, which is equal to the number of sources.

With power control, we observe a reduction in the schedule length in Fig. 6(b) as some of the interfering links are eliminated, thus increasing slot re-usability. When $\alpha = 3.0$, most of the interference can be eliminated by power control, and beyond which the structure of the routing tree, especially the number of nodes n_k on the largest branch with $(2n_k - 1) > N$, becomes the bottleneck. However, for $\alpha \geq 3.5$, power control cannot always eliminate interference as networks gets sparser and nodes tend to transmit at their maximum power.

7.2 Impact of Multi-Channel Scheduling

In this section, we analyze the performance of the channel assignment methods discussed in Section 5.2. We use CC2420 radios that have 16 channels in the 2.4 GHz range, with adjacent channels overlapping according to the rejection and blocking values given in the data sheet. We assume that the nodes transmit at maximum power and use minimum-hop trees. In TMCP and RBCA, time slots are assigned according to BFS-TIMESLOTASSIGNMENT for aggregated convergecast and LOCAL-TIMESLOTASSIGNMENT for raw data convergecast. The path loss exponent is taken as 3.5.

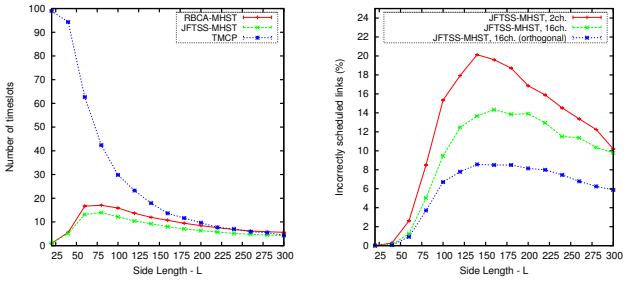
7.2.1 Aggregated Convergecast

Comparing the plots in Fig. 7(a) and Fig. 6(a), we observe that the channel assignment methods achieve schedule lengths that are shorter than those achieved by power control. While it's true that power control helps in reducing the effects of interference, this gain is limited due to the discrete levels and limited range of transmission power (e.g., CC2420 has 8 different power levels between 0 dBm and -24 dBm). In sparse deployments, nodes cannot reduce their transmit power below a certain threshold because a transceiver cannot decode signals below the sensitivity level (-95dBm). As shown in Fig. 6(a), for L $>$ 200, the schedule lengths are similar when nodes transmit at maximum power or when they adjust their power levels. Moreover, in mid-sparse deployments($60 \leq L \leq 180$, Fig. 6(a)) the limited range and discrete power levels restrict the nodes to adjust their transmit powers. On the other hand, multi-channel communication even with just two frequencies (Fig. 7(a)), can eliminate the interference limitations, and beyond this, the performance gains are limited by the connectivity structure. By transmitting on different channels, interference is eliminated by the high adjacent/alternate channel rejection values of the C2420 radio, and the channels behave like orthogonal.

In Fig 7(a), sparser deployments ($L > 140$) with multi-channel communication show a 40% reduction in schedule length as compared to transmitting on a single channel with maximum power. However, in denser deployments, multiple channels do not help much due to increased connectivity, with the sink as a bottleneck in the densest setting. From Fig 7(a), we observe that JFTSS and RBCA can optimally schedule the network using 16 channels, i.e., they achieve the lower bound, as shown by the line "Lower Bound-MHST". The advantage of RBCA over JFTSS is that it takes into account the topological characteristics: a parent node receives data on the same channel from its children, and does not have to switch channels in every slot. In dense deployments, TMCP performs better due to the different routing trees constructed, i.e., when $L = 20$, RBCA and JFTSS construct a star-topology, whereas TMCP constructs a 2-branch tree with 2 channels, a 16-branch tree with 16 channels.

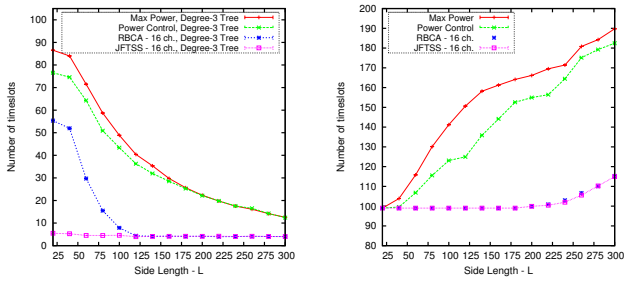
7.2.2 Raw Data Convergecast

In Fig. 7(b), we observe that none of the methods can eliminate interference completely with 2 channels, however, JFTSS and RBCA can do so with 6 or more channels at different densities (plots are not shown due to lack of space). We also see that TMCP needs 16 channels to reach a performance similar to that achieved by RBCA and JFTSS with only 2 channels. This is because in JFTSS and RBCA, when a node is receiving from its children, its parent can transmit simultaneously on a different channel, which is not possible due to intra-branch interference in TMCP. The results also verify that JFTSS and RBCA can achieve a schedule length which is



(a) (b)

Fig. 8: (a) Bounds on the number of frequencies. (b) Percentage of incorrectly scheduled links.



(a) (b)

Fig. 9: (a) Scheduling on degree-constrained minimum-hop trees. (b) Scheduling on CMST.

bounded by $\max(2n_k - 1, N)$, shown as "Lower Bound-MHST", so long as the number of available channels is sufficient to eliminate interference. Compared to the results on a single channel in sparser scenarios, we achieve a reduction of up to 40% on the schedule length. In very dense scenarios, the improvement is small because most of the nodes can directly reach the sink, and so the limiting factor becomes the half-duplex transceiver.

7.2.3 Required Number of Channels

In this section, for the different channel assignment methods, we evaluate the required number of channels to completely eliminate interference as a function of deployment density. In our simulation results, as shown in Fig. 8(a), we assume that the number of available channels is unlimited so as to show the upper bounds.

With RBCA and JFTSS, the number of channels required is low for dense networks as the number of receivers is low. In particular, when $L = 20$, all the nodes can directly connect to the sink, and so only one frequency is needed. As the network gets sparser, the number of receivers increases, and thus more frequencies are required to support concurrent transmissions. However, for $L \geq 80$, the number of nodes that are being connected to the same parent slowly dominates the effect of the number of receivers, and since the network gets very sparse, the number of channels required further goes down as the level of interference decreases.

The trends of both RBCA and JFTSS are quite similar, and the number of channels required is no more than the number of available channels on CC2420 radios (16 channels). On the other hand, TMCP requires many more channels as each branch is on a different channel. This is expensive for deployments where a lot of nodes can directly connect to the sink, and thus are assigned different channels because they form different branches. Thus, one needs to optimize the channel usage.

7.2.4 Interference Models and Orthogonal Channels

We now evaluate the impact of interference and channel models on the schedules, in particular, the idealistic assumption of the protocol model and the orthogonality assumption. We examine the feasibility of the schedules based on the adjacent and alternate channel rejection values of the transceivers and the SINR threshold.

Fig. 8(b) shows the results for JFTSS in terms of the percentage of nodes that are incorrectly scheduled (henceforth, referred to as errors). The top two lines show the errors for 2 and 16 channels with both the assumptions, whereas the bottom line shows the errors only for the orthogonality assumption. We observe that the errors are much higher in sparser deployments, because although the interference created by an individual sender is not high enough to jam concurrent transmissions, the cumulative effect from multiple senders is very high, which is not captured in the protocol model. On the other hand, in dense deployments, a single transmitter can jam another one because of smaller inter-node distances and higher level of interference. In such cases, some of the nodes might select the next available channels for concurrent transmissions, however, interference can still be high because the channels in reality are not perfectly orthogonal. After the peak, the network gets sparser and interference reduces. We note that, our simulations corroborate previous results [27] that the protocol model may result in serious interference.

7.3 Impact of Routing Trees

In the preceding sections, we observed that although interference can be substantially eliminated by using power control and multiple channels, connectivity of the tree still limits the performance. In the following, we discuss the improvements with routing tress.

7.3.1 Aggregated Convergecast on Degree-Constrained Trees

Fig. 9(a) shows the variation of schedule length with density when the maximum tree degree is 3 (in sparser scenarios, with a maximum degree of 2, it was not always possible to construct connected topologies). The top two lines are for nodes transmitting at maximum power, and nodes using power control. The bottom two lines are for JFTSS and RBCA. When nodes transmit with maximum power, we observe a reduction in the schedule length in dense deployments as compared to non-degree constrained trees shown in Fig. 6(a). We also notice further improvement with power control in denser deployments than in sparser ones.

When nodes are assigned channels using RBCA, we see a factor of more than 2 reduction in the schedule length in dense deployments ($L < 120$), as compared to

that using RBCA on minimum-hop spanning trees. We also observe that the schedule lengths are much larger than the maximum degree in the routing tree for dense deployments, as compared to those in sparse scenarios ($L \geq 120$). Considering deployments at different densities, routing over minimum-hop degree-constrained spanning trees together with RBCA achieves an order of magnitude improvement than routing over minimum-hop spanning trees while transmitting at maximum power. When we use JFTSS, the schedule length is close or equal to the maximum degree since it can handle interference using multiple channels more effectively by reusing and assigning them to the links instead of the receivers.

7.3.2 Raw Data Convergecast on CMST

Fig. 9(b) shows the variation of schedule length on CMST. The impact of such routing trees is more prominent in sparser networks ($L \geq 200$) than routing over minimum-hop spanning trees (Fig. 7(b)). When $L < 200$, the length is bounded by N. Beyond this point, it is almost always not possible to construct trees where the constraint $2n_k - 1 < N$ holds. In such cases, the schedule length is limited by $2n_k - 1$. These results indicate that RBCA and JFTSS combined with a suitable tree construction mechanism can achieve a reduction of up to 50% in the schedule length as compared to single-channel communication on minimum-hop spanning trees.

8 Conclusions

In this paper, we studied fast convergecast in WSN where nodes communicate using a TDMA protocol to minimize the schedule length. We addressed the fundamental limitations due to interference and half-duplex transceivers on the nodes and explored techniques to overcome the same. We found that while transmission power control helps in reducing the schedule length, multiple channels are more effective. We also observed that node-based (RBCA) and link-based (JFTSS) channel assignment schemes are more efficient in terms of eliminating interference as compared to assigning different channels on different branches of the tree (TMCP).

Once interference is completely eliminated, we proved that with half-duplex radios the achievable schedule length is lower-bounded by the maximum degree in the routing tree for aggregated convergecast, and by $\max(2n_k - 1, N)$ for raw-data convergecast. Using optimal convergecast scheduling algorithms, we showed that the lower bounds are achievable once a suitable routing scheme is used. Through extensive simulations, we demonstrated up to an order of magnitude reduction in the schedule length for aggregated, and a 50% reduction for raw-data convergecast. In future, we will explore scenarios with variable amounts of data and implement and evaluate the combination of the schemes considered.

References

[1] S. Gandham, Y. Zhang and Q. Huang, "Distributed Time-Optimal Scheduling for Convergecast in Wireless Sensor Networks", *Computer Networks*, vol. 52, nr. 3, 2008, pp. 610–629.

[2] K.K. Chintalapudi and L. Venkatraman, "On the Design of MAC Protocols for Low-Latency Hard Real-Time Discrete Control Applications over 802.15.4 Hardware", in *IPSN '08*, pp. 356–367.

[3] I. Talzi, A. Hasler, G. Stephan and C. Tschudin, "PermaSense: investigating permafrost with a WSN in the Swiss Alps", in *EmNets '07*, pp. 8–12.

[4] S. Upadhyayula and S.K.S. Gupta, "Spanning tree based algorithms for low latency and energy efficient data aggregation enhanced convergecast (DAC) in wireless sensor networks", *Ad Hoc Networks*, vol. 5, nr.5, 2007, pp. 626–648.

[5] T. Moscibroda, "The worst-case capacity of wireless sensor networks", in *IPSN '07*, pp. 1–10.

[6] T.ElBatt and A. Ephremides, "Joint Scheduling and Power Control for Wireless Ad-hoc Networks", in *INFOCOM '02*, pp. 976–984.

[7] Ö. Durmaz Incel and B. Krishnamachari, "Enhancing the Data Collection Rate of Tree-Based Aggregation in Wireless Sensor Networks", in *SECON '08*, pp. 569–577.

[8] 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.

[9] A. Ghosh, Ö. Durmaz Incel, V.A. Kumar and B. Krishnamachari, "Multi-Channel Scheduling Algorithms for Fast Aggregated Convergecast in Sensor Networks", in *MASS '09*, pp. 363–372.

[10] V. Annamalai, S.K.S. Gupta, L. Schwiebert, "On tree-based convergecasting in wireless sensor networks", in *WCNC '03*, pp. 1942–1947.

[11] X. Chen, X. Hu and J. Zhu, "Minimum Data Aggregation Time Problem in Wireless Sensor Networks", in *MSN '05*, pp. 133–142.

[12] W. Song, F. Yuan and R. LaHusen, "Time-Optimum Packet Scheduling for Many-to-One Routing in Wireless Sensor Networks", in *MASS '06*, pp. 81–90.

[13] H. Choi, J. Wang and E. Hughes, "Scheduling for Information Gathering on Sensor Network", *Wireless Networks (Online)*, 2007.

[14] N. Lai, C.King and C. Lin, "On Maximizing the Throughput of Convergecast in Wireless Sensor Networks", in *GPC '08*, pp. 396.

[15] M. Pan and Y. Tseng, "Quick convergecast in ZigBee beacon-enabled tree-based wireless sensor networks", *Computer Communications*, vol. 31, nr. 5, 2008, pp. 999-1011.

[16] W. Song, H. Renjie, B. Shirazi, R. LaHusen, "TreeMAC: Localized TDMA MAC protocol for real-time high-data-rate sensor networks", *Pervasive Mob. Comput.*, vol. 5, nr. 6, 2009, pp. 750.

[17] G. Zhou, C. Huang, T. Yan, T. He, J. Stankovic and T. Abdelzaher, "MMSN: Multi-Frequency Media Access Control for Wireless Sensor Networks", in *INFOCOM '06*, pp. 1–13.

[18] Y. Kim, H. Shin, H. Cha, "Y-MAC: An Energy-Efficient Multi-channel MAC Protocol for Dense Wireless Sensor Networks", in *IPSN '08*, pp. 53–63.

[19] B. Krishnamachari, D. Estrin and S.B. Wicker, "The Impact of Data Aggregation in Sensor Networks", in *ICDCSW '02*, pp. 575–578.

[20] J. Zander, "Performance of optimum transmitter power control in cellular radio systems", *IEEE Transactions on Vehicular Technology*, vol. 41, nr. 1, 1992, pp. 57–62.

[21] P. Kyasanur and N. H. Vaidya, "Capacity of multi-channel wireless networks: impact of number of channels and interfaces", in *MobiCom '05*, pp. 43–57.

[22] G. Sharma, R.R. Mazumdar and N.B. Shroff, "On the complexity of scheduling in wireless networks", in *MobiCom '06*, pp. 227–238.

[23] 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.

[24] C.H. Papadimitriou, "The complexity of the capacitated tree problem", *Networks*, vol. 8, nr. 3, 1978, pp. 217-230.

[25] H. Dai and R. Han, "A node-centric load balancing algorithm for wireless sensor networks", in *GLOBECOM '03*, pp. 548–552.

[26] M. Zuniga and B. Krishnamachari, "An analysis of unreliability and asymmetry in low-power wireless links", *ACM Trans. Sen. Netw.*, vol. 3, nr. 2, 2007, pp. 7.

[27] J. Grönkvist and A. Hansson, "Comparison between graph-based and interference-based STDMA scheduling", in *MobiHoc '01*, pp. 255–258.

Özlem Durmaz Incel is currently a postdoctoral researcher in the Networking Laboratory of Bogazici University, Turkey. She received her Ph.D. in Computer Science from the University of Twente, Netherlands, in March 2009. Her dissertation focused on efficient data collection in wireless sensor networks. She was a visiting student in the Autonomous Networks Research Group of the University of Southern California as part of her Ph.D. studies in 2007 and 2008.

Amitabha Ghosh is currently a postdoctoral researcher in Princeton University. He received his Ph.D. in Electrical Engineering from the University of Southern California in August 2010. His research interest is on design and analysis of algorithms for scheduling, routing, and power control for cellular and wireless sensor 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.

Bhaskar Krishnamachari is an Associate Professor in the Ming Hsieh Electrical Engineering department at the USC Viterbi School of Engineering. He received his B.E. in Electrical Engineering from The Cooper Union in 1998, and his M.S. and Ph.D. in Electrical Engineering from Cornell University in 1999 and 2002 respectively. His research interests are in design and performance analysis of protocols for next-generation wireless networks. He received the NSF CAREER award in 2004.

Krishnakant Chintalapudi is currently working as an associate researcher at Microsoft Research, India. His research interests include delay-sensitive and fault-tolerant wireless sensor networks, high-data-rate wireless sensor network applications, hybrid sensor networks, and wireless control and automation. Chintalapudi has a Ph.D. in Computer Science from the University of Southern California and an M.S. in Electrical Engineering from Drexel University.