International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

# Enhancing Data Collection of Tree-Based Aggregation in Wireless Sensor Networks

**B. Srikanth[1], K. Raghavendra Rao[2]**

[1]M.Tech Student, Department of CSE,
Anurag Group of Institutions, Hyderabad, India

[2]Assistant Professor, Department of CSE,
Anurag Group of Institutions, Hyderabad, India

**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 estimate a number of different techniques using realistic simulation models under the many-to-one communication paradigm known as convergecast. We first take time scheduling on a single frequency channel with the aim of minimizing the number of time slots required (schedule length) to complete a convergecast. Then, we combine scheduling with transmission power control to mitigate the effects of interference, and demonstrate that while power control helps in reducing the schedule length under a single frequency, and scheduling transmissions using multiple frequencies is more efficient. We provide lower bounds on the schedule length when interference is completely removed, and propose algorithms that achieve these bounds. We also calculate the performance of various channel assignment methods and find empirically that for moderate size networks of regarding 100 nodes, the use of multi-frequency scheduling can suffice to eliminate most of the interference. After that, the data collection rate no longer remains limited by interference but by the topology of the routing tree. Finally, we construct degree-constrained spanning trees and capacitated minimal spanning trees, and show significant development in scheduling performance over different deployment densities.***

**Keywords:** multiple channels, TDMA scheduling, 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 basic operation in wireless sensor networks (WSNs) [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 example, in safety and mission-critical applications where sensor nodes are deployed to detect oil/gas leak or structural damage, the actuators and controllers require receiving data from all the sensors within a specific deadline [2], failure of which might lead to unpredictable and catastrophic events. This falls under the type of one-shot data collection. Alternatively, applications such as permafrost monitoring [3] require periodic and fast data delivery over long periods of time, which falls under the type of continuous data collection. In this paper, we take 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 learn two types of data collection: 1) aggregated convergecast where packets are aggregated at each hop, with 2) raw-data convergecast where packets are individually relayed toward the sink. Aggregated convergecast is valid when a strong spatial correlation exists in the data, or the objective is to collect summarized information such as the maximum sensor reading. Raw-data convergecast, then again, is valid when every sensor reading is equally important, or the correlation is minimal. We learn aggregated convergecast in the context of continuous data collection, raw-data convergecast for one-shot data collection. These two types match up 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 considered, 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 Time Division Multiple Access (TDMA) are better fit for fast data collection, from the time when 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. Then, 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 attainable 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 is: 1) interference in the wireless medium, then 2) half-duplex transceivers on the sensor nodes, and 3) topology of the network. After that, we look at a number of different techniques that provide a hierarchy of successive improvements, the simplest along with which is an interference aware, minimum-length TDMA scheduling that enables spatial reuse. To achieve extra improvement, we combine transmission power control with scheduling, and apply multiple frequency channels to enable more concurrent transmissions. We explain that once multiple frequencies are employed along with spatial-reuse TDMA, the data collection rate regularly no longer remains limited by interference but by the topology of the network. So, in the

final step, we construct network topologies with specific properties that help in further enhancing the rate. Our main conclusion is that combining these different techniques can provide an order of magnitude improvement for aggregated convergecast, as well as a factor of 2 improvements for raw-data convergecast, compared to single-channel TDMA scheduling on minimum-hop routing trees.

Although the techniques of transmission power control and multichannel scheduling have been well studied for eliminating interference in general wireless networks, their performances for bounding the finishing point of data collection in WSNs have not been explored in detail in the previous studies. The basic novelty of our approach lies in the extensive exploration of the efficiency of transmission power control and multichannel communication on achieving fast convergecast operations in WSNs. also, we evaluate the impact of routing trees on fast data collection and to the best of our knowledge, as per previous studies, some of the existing work had the objective of minimizing the completion time of convergecasts. But, none of the previous work discussed the effect of multichannel 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 signify two extreme cases of data collection and together.

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

- **Bounds on convergecast scheduling**. We illustrate that if all interfering links are removed, the schedule length for aggregated convergecast is lower bounded by the maximum node degree in the routing tree, and for raw-data convergecast by maxð2nk _ 1;NÞ, where nk is the maximum number of nodes on any branch in the tree, and N is the number of source nodes. And we then begin 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 offer an unbounded improvement in the asymptotic capacity of aggregated convergecast. In this task, 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 get that for moderate size networks of 100 nodes, power control can decrease the schedule length by 15-20 percent.
- **Evaluation of channel assignment methods**. By extensive simulations, we show that scheduling transmissions on different frequency channels is more effective in mitigating interference as compared to transmission power control. We calculate the performance of three different channel assignment methods: 1) Joint Frequency Time Slot Scheduling (JFTSS), 2) Receiver-Based Channel Assignment [7], and 3) Tree-Based Multichannel Protocol (TMCP) [8]. These methods consider the channel assignment problem at different levels: the link level, node level, or cluster level. We demonstrate that for aggregated convergecast, TMCP performs improved than JFTSS and RBCA on minimum-hop routing trees, though performs worse on degree-constrained trees. In support of raw-data convergecast, RBCA and JFTSS perform better than TMCP, as 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 confirm that for aggregated convergecast, the performance can be better 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 converge cast, multichannel scheduling on capacitated minimal spanning trees (CMSTs) can reduce the schedule length by 50 percent.
- **Impact of channel models and interference**. In the setting of multiple frequencies, one simplifying assumption repeatedly made is that the frequencies are orthogonal to each other. We find this assumption and show that the schedules generated may not always eliminate interference, thus causing substantial packet losses. We also estimate and compare the two most commonly used interference models: 1) the graph-based protocol model, also 2) the Signal-to-Interference-plus-Noise Ratio (SINR)-based physical model.

## 2. Modeling and Problem Formulation

We model the multihop WSN as a graph $G=(V,E)$, where V is the set of nodes, and $E=\{(i,j)\mid i,j\in V\}$ is the set of edges representing the wireless links. A designated node $s\in V$ denotes the sink. The euclidean distance among two nodes i and j is denoted by $d_{ij}$. All the nodes except s are sources, which create packets and transmit them over a routing tree to s. We indicate the spanning tree on G rooted at s by $T=(V,E_T)$, where $E_T\subseteq E$ represents the tree edges. Every node is assumed to be equipped with a single half-duplex transceiver, which stops it from sending and receiving packets concurrently. We consider a TDMA protocol where time is divided into slots, in addition to consecutive slots are grouped into equal-sized nonoverlapping 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 guess that the interference range of a node is equal to its transmission range, i.e., two links cannot be scheduled at once if the receiver of at least one link is within the range of the transmitter of the other link. In the physical model, the unbeaten 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. So, a packet is received successfully at j if the

signal-to-interference-plus-noise ratio, $SINR_{ij}$, is greater than a certain threshold $\beta$, 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, N is the ambient noise level, and $g_{ij}$ is the propagation attenuation (link gain) between i and j. We use a easy distance dependent path-loss model to calculate the link gains as $g_{ij} = d_{ij}^{-\alpha}$, where the path-loss exponent $\alpha$ is a constant between 2 and 6, whose exact value depends on external circumstances of the medium (humidity, obstacles, etc.), as well as the sender-receiver distance. We suppose that the level of interference is static and does not change over time. For simplicity and ease of picture, 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 each frame, and raw-data convegecast for one-shot data collection where each node has only one packet to send. We suppose that the size of each packet is constant. Our objective is to deliver these packets to the sink over the routing tree as fast as possible. More specially, 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 $(i,j) \in E_T$ and $(k,l) \in E_T$ cannot be scheduled in the same time slot if they are adjacent to each other, i.e., if $\{i,j\} \cap \{k,l\} \neq \phi$. This constraint is due to the halfduplex 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 2- 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 $\beta$.

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 repetitive over multiple frames. So, a pipeline is established after a certain frame, and then forward to the sink continues to receive aggregated packets from all the source nodes once per frame. We give details about the pipelining in the next section. Conversely, 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 conditions link scheduling and node scheduling interchangeably as they are equivalent in our case. Note that the two other situations, which we do not believe in this paper due to space constraints, are one-shot aggregated convergecast with 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 since 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 finally established. Though, in the case of one-shot data collection, a node needs to stay for data from its children before being scheduled, which we pass on to as the causality constraint.

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

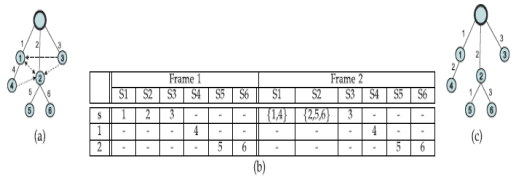
## 3. TDMA Scheduling of Convergecasts

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

### 3.1 Periodic Aggregated Convergecast

In this section, we consider the scheduling problem where packets are aggregated. Data aggregation is a normally used technique in WSN that can eliminate redundancy and minimize the number of transmissions, so 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., every 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 volume of aggregated data transmitted by each node is constant and does not depend on the size of the raw sensor readings. Typical instance of such aggregation functions are MAX ,MIN, MEDIAN, COUNT, AVERAGE, etc.




| | 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 |

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.

In Figs. 1a and 1b, we illustrate the notion of pipelining in aggregated convergecast and that of a schedule length on a network of six source nodes. The solid lines signify tree

edges, and the dotted lines stand for interfering links. The numbers next to the links represent the time slots at which the links are scheduled to transmit, and the numbers inside the circles indicate node ids. The entries in the table record the nodes from which packets are received by their corresponding receivers in each time slot. We make a note of that at the end of frame 1, the sink do not have packets from nodes 5 and 6; though, as the schedule is repeated, it receives aggregated packets from 2, 5, and 6 in slot 2 of the next frame. Also, 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, in that order. Therefore, a pipeline is established from frame 2, with the sink continues to receive aggregated packets from all the nodes once every six time slots. So, the minimum schedule length is 6.

### 3.1.1 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. Even if the problem of minimizing the schedule length is NP-complete on general graphs, we show in the following that once interference is removed, the problem reduces to 1 on a tree, and can be solved in polynomial time. At the end, we first give a lower bound on the schedule length, and then plan a time slot assignment scheme that achieves the bound.

**Lemma 1**. If all the interfering links are removed, 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 removed, the scheduling problem reduces to 1 on a tree. Now as each of the tree edges needs to be scheduled only once within each frame, it is equal to edge coloring on a graph, which requires number of colors at least equal to the maximum node degree.

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

### 3.1.2 Assignment of Time Slots

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.

**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**

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 allocated the minimum time slot that is different from all its adjacent edges respecting interfering constraints. Since we evaluate the performance of this algorithm also for the case when the interfering links are present, we check for the equivalent constraint in line 4; though, 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. We show the same network of Fig. 1a in Fig. 1c with all the interfering links eliminated, also the network is scheduled in three time slots.

Although BFS-TIMESLOTASSIGNMENT may not be an approximation to ideal scheduling under the physical interference model, it is a heuristic that can attain the lower bound if all the interfering links are eliminated. So, together with a method to eliminate interference, the algorithm can optimally plan the network.

**Theorem 1**. If all the interfering links are removed, the schedule length for aggregated convergecast achieved by BFSTIMESLOTASSIGNMENT is the minimum, i.e., $\Delta(T)$.

**Proof.** The proof is by induction on i. Let $T^i = (V^i, E_T^i)$ denote the subtree of T in the ith iteration constructed in the BFS order, where $E_T^i$ comprises all the edges that are assigned a slot, and $V^i$comprises the set of nodes on which the edges in$E_T^i$ are incident. Note that, $|E_T^i| = i$, because at every iteration, just 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 $(i+1)$st 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 1. Therefore,

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

If it remains the same, $N(i+1)$is still the maximum degree of $T^{i+1}$ at end of $(i+1)$stiteration. Otherwise, if it raise by 1, 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 ith iteration was $\Delta(T^i)$. This is so because in the BFS traversal, all the edges incident on a node are allocate a slot first before moving on to the next node, and as 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 ith iteration. Thus, at the end of $(i+1)$stiteration, 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. so, 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.

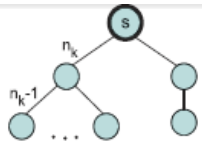



Fig. 2. Raw-data convergecast: largest top-subtree with $n_k$ nodes.

### 3.2 One-Shot Raw-Data Convergecast

In this section, we consider one-shot data collection where every sensor reading is equally essential, and so aggregation may not be desirable or even possible. Therefore, each of the packets has to be individually scheduled at each hop en route to the sink. As earlier, we focus on minimizing the schedule length. Unlike in the container of periodic aggregated convergecast where a pipelining takes place and each of the tree edges is scheduled only once within each frame, now 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 initially 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 example, 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 $\{3,7\}$.

#### 3.2.1 Lower Bound on Schedule Length

As mentioned in Section 3.1.1, if all the interfering links are eliminated using multiple frequencies, the simply limiting factor in minimizing the schedule length is the half-duplex transceivers. In the next, we give a lower bound on the schedule length under this scenario.

**Lemma 2**. If all the interfering links are removed, 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, plus 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 nonincreasing order of their sizes: $n_k \geq n_{k-1} \geq \cdots \geq n_1$. Consider the routing tree shown in Fig. 2. given that the nodes cannot receive multiple packets at the same time, N is a trivial lower bound to receive all the packets. Next, consider the largest topsubtree k, the root of which has to transmit nk packets to the sink, with the children of this root have to forward $n_k - 1$packets in total. As the half-duplex transceivers, time slots allocated to the root of this topsubtree 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 calculated by Gandham et al. [1] for the 2-hop interference model. They anticipated a time slot assignment scheme for tree networks, which requires each node to keep a buffer that stores at most two packets and minimizes the schedule length. In the following, we explain 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 storing more than one packet in buffers at any time.

#### 3.2.2 Assignment of Time Slots

We now describe a time slot assignment scheme in Algorithm 2, called LOCAL-TIMESLOTASSIGNMENT, and which is run locally by each node at each time slot. The key idea is to: 1) schedule transmissions in parallel along multiple branches of the tree, as well as 2) keep the sink busy in receiving packets for as many time slots as possible. Since the sink can receive from the root of at most one top-subtree in any time slot, we need to choose which top-subtree should be made active. We guess that the sink is aware of the number of nodes in every top-subtree. Each source node maintains a buffer and its associated state, which can be both full and empty depending on whether it contains a packet or not. Our algorithm does not need any of the nodes to store more than one packet in their buffer at any time. We start all the buffers as full, and suppose 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 topsubtrees. Wedefine a top-subtree to be eligible if its root has at least one packet to transmit. For a specified time slot, we schedule the root of an eligible top-subtree which has the largest number of total (remaining) packets. If none of the topsubtrees are suitable, 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 describe 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 in addition to the subtree rooted at this node is active, we schedule one of its children at chance whose buffer is not vacant. 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 when a node empties its buffer, and it will

receive a packet in the next time slot, so emptying buffers from the bottom of the subtree to the top.

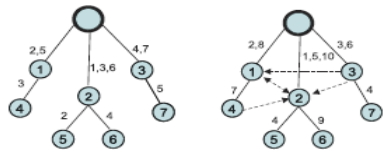



Fig. 3. Raw-data convergecast using algorithm LOCAL-TIMESLOTAS-SIGNMENT: (a) Schedule length of 7 when all the interfering links are removed. (b) Schedule length of 10 when the interfering links are present.

We run through an example shown in Fig. 3a to give details the algorithm. In the first time slot, since the eligible topsubtree 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 two 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 two remaining packets. We decide the first one at random and schedule the link $(2,s)$ 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 awaiting all the packets are delivered to the sink, yielding an task that requires seven time slots. Note that, in this example, $2n_k-1=5$, and so $\max(2n_k-1,N)=7$. In Fig. 3b, 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. previous to giving the details of the proof, we initially highlight the two key insights of the algorithm: 1) the sink is kept busy in receiving packets for as many time slots as possible, as well as 2) a node's buffer is not empty for two or more consecutive time slots so long as the subtree rooted at this node is active. The first one is obvious from the scheduling rule between the sink and the top-subtrees. We prove the second imminent 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. and 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 no increasing order of their sizes: $n_k \geq n_{k-1} \geq \cdots \geq n_1$. 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. So, the schedule length is at most $2n_k-1$.

Now suppose $n_k > \sum_{i=1}^{k-1} n_i$; 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 collect packets in every slot, because or else it remains idle during some time slots and the first condition $n_k > \sum_{i=1}^{k-1} n_i$will be met. So, we will prove that the algorithm keeps the inequality $n_k > \sum_{i=1}^{k-1} n_i$ 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 $n_k \geq n_{k-1} \geq \cdots \geq n_1$. 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 new top-subtrees with an equal number of packets left as k in slot t. Next, the root of a new largest topsubtree j is chosen, and the inequality holds since $n_j - 1 \leq \sum_{i=1}^{k-1} n_i - 1$ (since $n_j = n_{k_i}$).

Thus, the algorithm keeps the inequality as an invariant, and there always be a top-sub tree 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 two packets left at two different top-subtrees that can be scheduled in alternate slots. Because this inequality holds for all the N steps, the sink for all time 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. So, the theorem follows.

## 4. 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, we focus on different methods to mitigate the effects of interference on the schedule length. Initially, we discuss the benefits of using transmission power control and explain the basics of a possible algorithm. So, we discuss the advantages of using multiple channels by considering three different channel assignment schemes.

### 4.1 Transmission Power Control

In wireless networks, excessive interference can be eliminated by using transmission power control [6], [20], to be exact. 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, since opposed to a continuous range where an unbounded improvement in the asymptotic capacity can be achieved by using a nonlinear power assignment [5]. We first clarify the basics of one particular algorithm.

The algorithm proposed by ElBatt and Ephremides [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 objective is to find a TDMA schedule that can support as many transmissions as possible in each time slot. It has two phases: 1) scheduling and 2) power control that are executed at every time slot. Primarily, the scheduling phase searches for a valid transmission schedule, i.e., largest subset of nodes, wherever no node is to transmit and receive at the same time, or to receive from multiple nodes at the same time. Then, in the given valid schedule, and 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 alter 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 must decrease its power and if it is below the threshold, it must increase its transmission power, within the accessible range of power levels on the radio. If all the nodes meet the interfering restraint, the algorithm proceeds with the schedule calculation for the next time slot. Then again, if the maximum number of iterations is reached and there are nodes which cannot meet the interfering restraint, 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 recurring until an admissible transmission scenario is found.

### 4.2 Multichannel Scheduling

Multichannel communication is an efficient method to eliminate interference by enabling concurrent transmissions over different frequencies [21]. Although typical WSN radios operate on a limited band width, and their operating frequencies can be adjusted, so 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 civilizing link bandwidth by consolidating frequencies. In this section, we give details 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 think the channel assignment problem at different levels: the link level (JFTSS) and node level (RBCA), or cluster level (TMCP).

#### 4.2.1 Joint Frequency Time Slot Scheduling

JFTSS offers a greedy joint solution for constructing a maximal schedule, such that a plan is said to be maximal if it meets the adjacency and interfering restraints, and no more links can be scheduled for concurrent transmissions on any time slot and channel without violating the constraints. Estimate bounds on JFTSS for single-channel systems and its comparison with multichannel 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 identical, such as in aggregated convergecast, the mainly constrained link is considered primary, 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 vacant schedule and first sorts the links according to the loads or constraints. The most loaded or controlled link in the first available slot-channel pair is scheduled first and added to the schedule. All the links that have an adjacency restraint with the scheduled link are excluded from the list of the links to be scheduled at a given slot. The links that do not contain 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 achievable. The algorithm continues to schedule the links according to the most loaded (or most

constrained) metric. While no more links can be scheduled for a specified slot, the scheduler continues with scheduling in the next slot. Fig. 4a shows the same tree given in Fig. 1a which is planned according to JFTSS where aggregated data are collected. JFTSS begins 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.

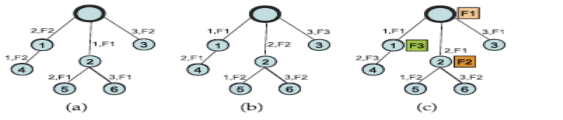



Fig. 4. Scheduling with multichannels for aggregated convergecast: (a) Schedule generated with JFTSS. (b) Schedule generated with TMCP. (c) Schedule generated with RBCA.

An advantage of JFTSS is that it is simple 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.

#### 4.2.2 Tree-Based Multichannel Protocol

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

#### 4.2.3 Receiver-Based Channel Assignment

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 achievable. In RBCA, the children of a common parent transmit on the same channel. Every node in the tree, thus, operates on at most two channels, so avoiding pairwise, per-packet channel negotiation expenses. The algorithm initially assigns the same channel to all the receivers. After that, 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). Though, due to adjacent channel overlaps, SINR values at the receivers might not always be high enough to tolerate interference, in which case the channels are allocated according to the ability of the transceivers to reject interference. We verified approximation factors for RBCA when used with greedy scheduling in [9]. Fig. 4c shows the same tree given in Fig. 1a scheduled with RBCA for aggregated convergecast. Initially, all nodes are on frequency F1. RBCA begin with the most interfered parent, node 2 in this case, and assigns F2. After that, it continues to assign F3 to node 3 as the second most interfered parent. As all interfering parents are assigned different frequencies, sink can get on F1.

## 5. Impact of Routing Trees

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

### 5.1 Aggregated Data Collection

We first construct balanced trees and compare their performance with unbalanced trees. We study that in both cases, the sink frequently creates a high-degree bottleneck. To conquer this, we then propose a heuristic, as explained in Algorithm 3, by modifying Dijkstra's shortest path algorithm to construct degree-constrained trees. Make a note of constructing such a degree-constrained tree is NP-hard. Every 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 $\infty$. The algorithm starts with the sink node, and adds a node $i' \notin T$ at each 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. So, in this latter situation, the heuristic may not always generate a spanning tree. We consider only those instances of the topologies where spanning trees with the specified degree constraint are produced.

**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**

To illustrate the gains of degree-constrained trees, regard as 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 choose the sink, and this will give a schedule length of N. Though, if we limit the number of children per node to 2, after that this will result in two sub trees rooted at the sink, and if there are sufficient frequencies to eliminate interference, and the network can be scheduled using only

two time slots, thus achieving a factor of N/2 reduction in the schedule length.

### 5.2 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 removed. 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 [24]. The CMST problem, which is known to be NP-complete, is to decide 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 container, the weight of each link is 1, and the prescribed capacity is $(N+\tilde{1})/2$. Here, we propose a heuristic, as described in Algorithm 4, based on the greedy system presented by Dai and Han [25], which solves a variant of the CMST problem by searching for routing trees with an equal number of nodes on each branch. We supplement their scheme with a new set of rules and grow the tree hop by hop outward from the sink. We imagine that the nodes know their minimum-hop counts to sink.

**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**

**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 advance processing, breaking ties based on the smallest id. We describe 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.

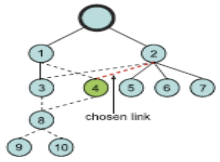



Fig. 5. Balanced tree construction: Node 4 is attached to $b2$ based on the search sets; load on both $b1$ and $b2$ is 5.

**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 precise 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 exacting, if the chosen node has access to branch b, and has a neighbor that can connect to only branch b if b is selected, after that this neighbor and its potential children are included in the search set for b. Though, if the neighbor has access to at least one other branch even after b is selected, after that it is not included in the search set.

The looks for sets guarantee that the choices for the nodes at longer hops to join a particular branch are not limited by the decision of the joining node. And this balances out the number of nodes on different branches and prevents one to grow faster than others. Formerly the search sets are constructed; we decide 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 condition shown in Fig. 5. Dotted lines signify potential communication links and solid lines represent already included tree edges. At this position, 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). So, the sum of the load and the size of the search set for b1 is 5, and that for b2 is 4. So, we connect 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.

## 6. Conclusion

In this paper, we studied fast convergecast in WSN where nodes communicate using a TDMA protocol to minimize the schedule length. We addressed the basic limitations due to interference and half-duplex transceivers on the nodes and explored techniques to overcome the same. We found that as transmission power control helps in reducing the schedule length, and 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 demonstrated that with half-duplex radios, the attainable 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 illustrated that the lower bounds are achievable once a suitable routing scheme is used. Through extensive simulations, we established up to an order of magnitude reduction in the schedule length for aggregated, with a 50 percent reduction for raw-data convergecast.

## References

[1] S. Gandham, Y. Zhang, and Q. Huang, "Distributed Time-Optimal Scheduling for Convergecast in Wireless Sensor Networks," Computer Networks, vol. 52, no. 3, pp. 610-629, 2008.
[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," Proc. Int'l Conf. Information Processing in Sensor Networks (IPSN '08), pp. 356-367, 2008.
[3] I. Talzi, A. Hasler, G. Stephan, and C. Tschudin, "PermaSense: Investigating Permafrost with a WSN in the Swiss Alps," Proc. Workshop Embedded Networked Sensors (EmNets '07), pp. 8-12, 2007.
[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, no. 5, pp. 626-648, 2007.
[5] T. Moscibroda, "The Worst-Case Capacity of Wireless Sensor Networks," Proc. Int'l Conf. Information Processing in Sensor Networks (IPSN '07), pp. 1-10, 2007.
[6] T. ElBatt and A. Ephremides, "Joint Scheduling and Power Control for Wireless Ad-Hoc Networks," Proc. IEEE INFOCOM, pp. 976-984, 2002.
[7] O. Durmaz Incel and B. Krishnamachari, "Enhancing the Data Collection Rate of Tree-Based Aggregation in Wireless Sensor Networks," Proc. Ann. IEEE Comm. Soc. Conf. Sensor, Mesh and Ad Hoc Comm. and Networks (SECON '08), pp. 569-577, 2008.
[8] Y. Wu, J.A. Stankovic, T. He, and S. Lin, "Realistic and Efficient Multi-Channel Communications in Wireless Sensor Networks," Proc. IEEE INFOCOM, pp. 1193-1201, 2008.
[9] A. Ghosh, O ¨. Durmaz Incel, V.A. Kumar, and B. Krishnamachari, "Multi-Channel Scheduling Algorithms for Fast Aggregated Convergecast in Sensor Networks," Proc. IEEE Int'l Conf. Mobile Adhoc and Sensor Systems (MASS '09), pp. 363-372, 2009.
[10] V. Annamalai, S.K.S. Gupta, and L. Schwiebert, "On Tree-Based Convergecasting in Wireless Sensor Networks," Proc. IEEE Wireless Comm. and Networking Conf. (WCNC '03), vol. 3, pp. 1942-1947, 2003.
[11] X. Chen, X. Hu, and J. Zhu, "Minimum Data Aggregation Time Problem in Wireless Sensor Networks," Proc. Int'l Conf. Mobile Ad- Hoc and Sensor Networks (MSN '05), pp. 133-142, 2005.
[12] W. Song, F. Yuan, and R. LaHusen, "Time-Optimum Packet Scheduling for Many-to-One Routing in Wireless Sensor Networks," Proc. IEEE Int'l Conf. Mobile Ad-Hoc and Sensor Systems(MASS '06), pp. 81-90, 2006.
[13] H. Choi, J. Wang, and E. Hughes, "Scheduling for Information Gathering on Sensor Network," Wireless Networks, vol. 15, pp. 127- 140, 2009.
[14] N. Lai, C. King, and C. Lin, "On Maximizing the Throughput of Convergecast in Wireless Sensor Networks," Proc. Int'l Conf. Advances in Grid and Pervasive Computing (GPC '08), pp. 396-408, 2008.
[15] M. Pan and Y. Tseng, "Quick Convergecast in ZigBee Beacon- Enabled Tree-Based Wireless Sensor Networks," Computer Comm., vol. 31, no. 5, pp. 999-1011, 2008.
[16] W. Song, H. Renjie, B. Shirazi, and R. LaHusen, "TreeMAC: Localized TDMA MAC Protocol for Real-Time High-Data-Rate Sensor Networks," J. Pervasive and Mobile Computing, vol. 5, no. 6, pp. 750-765, 2009.
[17] G. Zhou, C. Huang, T. Yan, T. He, J. Stankovic, and T. Abdelzaher, "MMSN: Multi-Frequency Media Access Control for Wireless Sensor Networks," Proc. IEEE INFOCOM, pp. 1-13, 2006.
[18] Y. Kim, H. Shin, and H. Cha, "Y-MAC: An Energy-Efficient Multi- Channel MAC Protocol for Dense Wireless Sensor Networks," Proc. Int'l Conf. Information Processing in Sensor Networks (IPSN '08), pp. 53-63, Apr. 2008.
[19] B. Krishnamachari, D. Estrin, and S.B. Wicker, "The Impact of Data Aggregation in Wireless Sensor Networks," Proc. Int'l Conf. Distributed Computing Systems Workshops (ICDCSW '02), pp. 575- 578, 2002.
[20] J. Zander, "Performance of Optimum Transmitter Power Control in Cellular Radio Systems," IEEE Trans. on Vehicular Technology, vol. 41, no. 1, pp. 57-62, Feb. 1992.
[21] P. Kyasanur and N.H. Vaidya, "Capacity of Multi-Channel Wireless Networks: Impact of Number of Channels and Interfaces," Proc. ACM MobiCom, pp. 43-57, 2005.
[22] G. Sharma, R.R. Mazumdar, and N.B. Shroff, "On the Complexity of Scheduling in Wireless Networks," Proc. ACM MobiCom, pp. 227-238, 2006.
[23] X. Lin and S. Rasool, "A Distributed Joint Channel-Assignment, Scheduling and Routing Algorithm for Multi-Channel Ad-Hoc Wireless Networks," Proc. IEEE INFOCOM, pp. 1118-1126, 2007.
[24] C.H. Papadimitriou, "The Complexity of the Capacitated Tree Problem," Networks, vol. 8, no. 3, pp. 217-230, 1978.
[25] H. Dai and R. Han, "A Node-Centric Load Balancing Algorithm for Wireless Sensor Networks," Proc. IEEE Conf. Global Telecomm. (GlobeCom '03), pp. 548-552, 2003.
[26] M. Zuniga and B. Krishnamachari, "An Analysis of Unreliability and Asymmetry in Low-Power Wireless Links," ACM Trans. Sensor Networks, vol. 3, no. 2, p. 7, 2007.
[27] J. Gro¨nkvist and A. Hansson, "Comparison between Graph-Based and Interference-Based STDMA Scheduling," Proc. ACM Mobi- Hoc, pp. 255-258, 2001.

## Author Profile

**B. Srikanth** received the B.Tech degree in Computer Science of Engineering from JNTU Hyderabad in 2012 and now pursuing M.Tech degree in Computer science from Anurag Group of Institutions from JNTU Hyderabad.

**K. Raghavendra Rao** M.TECH working as assistant professor in Computer Science Engineering from CVSR College Of Engineering from Anurag Group of Institutions Venkatapur (V), Ghatkesar (M), Ranga Reddy District, Hyderabad-500088, Telangana State.