This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2015.2512272, IEEE Transactions on Wireless Communications

On the Interaction between Scheduling and Compressive Data Gathering in Wireless Sensor Networks

Dariush Ebrahimi, Chadi Assi, *Senior Member, IEEE,*

***Abstract*—Compressive data gathering (CDG) has emerged as a useful method for collecting sensory data in large scale sensor networks; this technique is able to reduce global scale communication cost without introducing intensive computation, and is capable of extending the lifetime of the entire sensor network by balancing the aggregation and forwarding load across the network. With CDG, multiple forwarding trees are constructed, each for aggregating a coded or compressed measurement, and these measurements are collected at the sink for recovering the uncoded transmissions from the sensors. This paper studies the problem of constructing forwarding trees for collecting and aggregating sensed data in the network under the realistic physical interference model. The problem of gathering tree construction and link scheduling is addressed jointly, through a mathematical formulation, and its complexity is underlined. Our objective is to collect data at the sink with both minimal latency and fewer transmissions. We show the joint problem is NP-hard and owing to its complexity, we present a decentralized method for solving the tree construction and the link scheduling sub-problems. Our link scheduling sub-problem relies on defining an interference neighbourhood for each link and coordinating transmissions among network links to control the interference. We prove the correctness of our algorithmic method and analyse its performance. Numerical results are presented to compare the performance of the decentralized solution with the joint model as well as prior work from the literature.**

***Index Terms*—Wireless Sensor Network, Compressive Data Gathering, Routing, SINR, Link Scheduling, Energy, Latency.**

I. Introduction

Wireless sensor networks (WSNs) have received significant attention due to their versatility and have been deployed widely in applications ranging from health monitoring to the monitoring of environment, traffic, underground water pipes, structural and critical infrastructure, among others. Many of these applications require sensors to periodically sense and send sensory data to a remote central unit (e.g., sink) for processing, often through multi-hop paths. These energy limited sensors, once deployed, may receive little or no maintenance, and therefore gathering data in the most energy efficient manner becomes critical for the longevity of wireless sensor networks. Compressive Data Gathering (CDG), based on compressive sensing (CS) theory, is one of the most efficient methods for gathering sensed data en-route to the sink [1] and has recently been receiving focal attention owing to its ability to reduce the global communication cost without incurring intensive computation or transmission overhead. With compressive data gathering, rather than receiving all readings, e.g., from n sensors, the sink will only receive few weighted (encoded) sums (e.g., m, $m \ll n$) of all the readings, from which the sink will be able to recover (decode) the original data, as long as the readings can be transformed or compressed in some sparse orthonormal transform domain [1,2]; here, $m = O(k\log n)$ and k represents the sparsity representation of the data in the transform domain. CDG has attracted researchers' attention only recently; this technique has shown to yield substantial energy savings, therefore extending the network lifetime, and achieve load balancing by dispersing the communication costs to all sensors along a given route [1].

In our work, we suppose the original data is compressible in some transform domain, and it is recovered at the sink by receiving m sparse projections [3], where each projection corresponds to an aggregation of data from sensors according to the theory of compressive sensing. Here, projections are gathered by establishing forwarding trees, one tree for each projection which gathers coded (compressed) data from nodes involved in the projection. Projections may be either collected by projection nodes (selected sensors), which subsequently send their collected coded measurements to the sink (e.g., through shortest paths) to recover the original data, or at the sink itself. Upon collecting all projections, the sink then attempts to recover the original data by solving a convex optimization problem [4]. In our work we suppose the sink directly collects compressed measurements, and therefore, our gathering problem reduces to constructing forwarding trees, each rooted at the sink, for efficiently collecting measurements.

In this paper, we consider the interaction between forwarding tree construction and link scheduling under the physical interference model. Namely, once the gathering trees are constructed, links on the constructed trees need to be scheduled for transmissions such that adjacent transmissions do not cause harmful interference on one another (thus corrupting the compressed measurements) while maintaining a maximum spatial reuse of the wireless spectrum. Further, unlike previous work (e.g. [5]), to reduce the number of transmissions along the forwarding trees, parent nodes should only transmit their measurements upon receiving measurements from their children; once downstream coded/compressed measurements are received, such measurements are combined with local measurements for uplink transmission. Therefore, finding forwarding trees to collect measurements at the sink in the

D. Ebrahimi and C. Assi are with the Faculty of Engineering and Computer Science, Concordia University, 1455 De Maisonneuve Blvd. W., Montreal, QC H3G1M8, Canada e-mail: darebra@yahoo.com, assi@ciise.concordia.ca.

This work is supported by Concordia University Research Chair and FQRNT Team Grant.

1536-1276 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2015.2512272, IEEE Transactions on Wireless Communications

most energy efficient manner under the physical interference model becomes a complex problem of combinatorial nature. We mathematically model this problem and underline its complexity. It should be noted here that link scheduling in wireless networks under the physical interference model is shown to be NP-complete [6,7]; we further show that the problem of scheduling links belonging to the forwarding trees is NP-hard. Owing to the complexity of the joint problem, we also present a decentralized method for solving the tree construction and the link scheduling sub-problems. Our link scheduling sub-problem relies on defining a local interference neighbourhood for each link and coordinating transmissions, through message exchange, among network links to control the level of interference in the neighbourhood of each link. Our routing subproblem consists of solving a tree construction subproblem in a decentralized way and adding refinements to help achieve better scheduling performance. More specifically, the main contributions of our paper are summarized as follows:

- We define the problem of forwarding tree construction and link scheduling (FTCS) and we mathematically formulate the problem as a mixed integer linear program (MILP) through which we may obtain optimal solutions for small size networks.
- We analyze the complexity of FTCS.
- To overcome the computational complexity, we propose a distributed method that can solve for large scale networks.
- We prove the correctness of our algorithmic method and analyze its performance.
- We validate through simulations the efficiency and performance of our distributed FTCS method.

To the best of our knowledge, our work is the first to resolve the problem of compressive data gathering under physical interference constraints in a decentralized manner without requiring to partition the unit square area into cells. Our method can be used to efficiently operate large wireless sensor networks which periodically gather sensory data in the most energy efficient manner and with gathering latency constraints. Our work is compared against a state of the art technique from the recent literature. The rest of the paper is organized as follows. Section II presents system model and problem description. A mathematical model is given in Section III. Section IV proposes the algorithmic method, followed by its performance analysis and numerical results in Section V and VI respectively. We summarize the related work in Section VII and conclude in section VIII.

II. System Model and Problem Definition

A. System Model

We model a wireless sensor network as a connected graph $G = (V, E)$, where V is the set of n nodes deployed randomly in a region and E is the set of links between any two sensor nodes which reside within each other's communication radius. The density of the network can be adjusted by varying the transmission power of the nodes. However, varying the transmit power yields a system model that is much harder to solve. For simplicity, we assume a fixed transmit power P for all sensor nodes and we assign the power P such that the resulting graph is connected without a single disconnected node. We assume each sensor at each round (period) has a data reading x_i (for example, speed, density or temperature) which it intends to send to the sink that may be located at a certain location in the network. Consequently, at each round, the sink needs to gather, in total, a data vector of size n ($X = [x_1, x_2, ..., x_n]^T$) from all the nodes in the network. Since not all the nodes may have a direct link with the sink, sensors will send their readings over multi hop routes. We consider a Time Division Multiple Access (TDMA) based MAC access where time is divided into slots of equal length; we define the set of links which can be active concurrently in the same time slot as a ***configuration***. Here, a configuration consists of links/transmissions from multiple forwarding trees which may be active simultaneously, such that no one parent (in one tree) is scheduled for transmission before it receives transmissions from its children. Let d_{ij} be the Euclidean distance between two nodes i and j and let G_{ij} be the channel gain from a transmitter node i to a receiver node j, (e.g., $G_{ij} = d_{ij}^{-\alpha}$, α is the path lost exponent). Now, under the physical interference model [8], in the presence of concurrent transmissions, a receiver j can successfully receive the transmission from node i if the signal to interference plus noise ratio (SINR) at j is above a certain threshold β, which is formulated as:

$$SINR_{(i,j)} = \frac{P\,G_{ij}}{\eta + \sum_{\forall (h,k)\in E: h\neq i} P\,G_{hj}} \geq \beta \qquad \forall (i,j) \in E \tag{1}$$

where η is the background noise. In general, we refer to the number of time slots needed to schedule the links in all forwarding trees (to collect all compressed measurements) as a round. The size of a round determines the latency for collecting the measurements. We further assume all packets (each carrying a compressed measurement) are of equal size.

B. Compressive Data Gathering

Compressive Data Gathering (CDG) promises to efficiently recover n sensors' readings at the sink with far fewer sample measurements, as long as the original readings could be transformed or compressed in some sparse orthonormal transform domain. Suppose the original data $X = [x_1, x_2, ..., x_n]^T$ has a k-sparse representation under a proper matrix Ψ, where Ψ is a Fourier transform matrix of size $n \times n$. That is $X = \Psi\hat{S}$, where $\hat{S}$ is a k-sparse column vector representation of X and only k coefficients of $\hat{S}$ are non-zero and $k \ll n$. According to the Restricted Isometry Property (RIP) of the CS theory [2], the sink may receive $m = O(k \log n)$ measurements instead of n readings, where $m \ll n$; that is $Z = \Phi\Psi\hat{S} = \Phi X$, where Z is a column vector of sample measurements of size $m \times 1$ and Φ is a random sample measurement matrix of size $m \times n$ (the column vectors of Φ are chosen at random with i.i.d entries from a normal distribution, or more generally they follow any Gaussian distribution). In other words, the sink can perfectly recover the original data X by receiving $Z = [z_1, z_2, ..., z_m]^T$, where $z_t = \sum_{j=1}^{n} \phi_{tj} x_j$, $t = 1, 2, ..., m$ and ϕ_{tj} is a coefficient in matrix Φ at row t and column j. Each z_t represents a weighted sum of measurements from nodes in the network

1536-1276 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2015.2512272, IEEE Transactions on Wireless Communications

with non-zero coefficients in a row of the matrix Φ. We refer to these nodes as interest nodes and the data aggregated from those interest nodes as one projection. The matrix Φ has m rows, one row for each weighted sum (projection), and n columns, one column for each sensor node.

Now, from m measurements (Z), using the random sample matrix Φ and the Fourier transform matrix Ψ, the sink recovers the sparse representation of the data $\tilde{S}$ (not the original data) by solving the convex optimization problem of (2).

$$\min_{\tilde{S}} \|\tilde{S}\|_1 \qquad subject\ to \qquad Z = \Phi\Psi\tilde{S} = A\tilde{S} \tag{2}$$

After recovering the sparse vector $\tilde{S}$, the original data (X) is obtained by letting $X = \Psi\tilde{S}$. For data recovery, the matrix $A = \Phi\Psi$ has to satisfy the RIP. A matrix A obeys the RIP with high probability if the entries are chosen according to a Gaussian, Bernoulli, or more generally, any sub-gaussian distribution [2]. Note that the matrix Ψ is only required at the sink for decoding (recovering) and it is not required for encoding at the nodes. Matrix Φ is fixed and can be considered as *a priori* knowledge for the entire network [1] or, each random vector (corresponding to one sensor node) can be generated locally at each node using a predetermined seed for a pseudo random generator. Seeds may be distributed by the sink to the nodes in the networks. For more details on CS, the reader is referred to [2,4].

In [3], the authors have shown that there is a trade-off between the sparsity of the projections (number of nonzero coefficients in each row of the matrix Φ) and the number of projections needed (number of rows in matrix Φ). In this paper, to distribute the non-zero coefficients more evenly in the matrix Φ and make each projection as sparse as possible, as in [9], the number of non-zero coefficients in each row of the matrix Φ is chosen as $\lceil \frac{n}{m} \rceil$ such that none of the columns in Φ has all-zero entries. Since the sparsity and number of projections (m) depend on the k-sparsity Fourier transform representation of sensors' readings (as we mentioned earlier), the random sample matrix Φ presented here satisfies all the conditions required to fully recover the original data readings at the sink using the compressive sensing technique.

C. Problem Description

We are interested in gathering, in each round, measurements at the sink from all the sensors. We assume sensors have finite battery lifetime. We also assume transmissions in the network can interfere with one another and therefore an access scheme should be in place to coordinate the transmissions.

Problem Definition 1 (Forwarding tree construction in PCDG): *Given a connected graph G of n sensor nodes, a sink, and a sparse matrix Φ, the problem of finding tree construction in projection based compressive data gathering (PCDG) consists of finding m forwarding trees, each tree to collect coded measurements from a subset of nodes (nodes with non-zero coefficients in a corresponding row of matrix Φ, where such nodes are referred to as interest nodes) en-route to the sink in the most energy efficient manner.*

Here, each tree t ($1 \leq t \leq m$) corresponds to one projection which gathers one weighted sum z_t from a set of interest nodes at the sink. Our objective is to construct these trees such that the total number of transmissions in the network is minimized.The gathering on each routing tree is performed as follows; each interest node (j, $j \in I_t$, where I_t is the set of interest nodes of tree t) upon collecting its measurement, multiplies its reading x_j with its random coefficient ϕ_{tj} and combines the data $\phi_{tj}x_j$ with those received from its descendants (if any) and sends the obtained weighted (coded) sum in one packet to the parent node. The sink which is the root of all trees will receive one weighted sum $z_t = \sum_{j=1}^{n} \phi_{tj}x_j$ from each set of interest nodes (i.e., tree). Without compressive data gathering, nodes closer to the sink will perform more forwarding than nodes farther away from the sink. Thus, with CDG, the transmission load is dispersed across all sensors in the network, and the energy consumption is balanced, yielding an extended network lifetime. It should be noted that this problem differs from plain compressive data gathering [1] in that it uses sparse random projections, and differs from distributed sparse random projections [3] since we allow in-network data gathering en-route to the sink.

Problem Definition 2 (Scheduling): *Given a set of forwarding trees, the scheduling problem consists of finding maximal size sets (where a set is a configuration*[1] *of active links in one time slot) and allocating time slots for them such that the resulting schedule length is minimized. Such problem guarantees the delivery of compressed measurements to the sink with minimal latency.*

Problem Definition 3 (FTCS): *The joint problem of forwarding tree construction and scheduling (FTCS) is the combination of problems 1 and 2.*

We illustrate the operation of FTCS on the sample network shown in Fig. 1; namely, we illustrate the interaction between the tree construction and link scheduling and highlight the impact on the data gathering latency (or the schedule length). We compare a joint FTCS method with one that constructs trees and schedule them separately. The results are depicted in Fig. 1(a)-1(b). The example shows how to gather data at the sink from all sensors using three projections. As the figures show, both methods require the same number of transmissions (links) to gather the data, however, Fig. 1(a) shows that the trees in the joint FTCS can be scheduled in only 8 time slots, whereas, the disjoint method, as Fig. 1(b) shows, requires 9 time slots to collect the measurements. This is due to the fact that trees are constructed without considering the requirements for achieving shorter schedule length. Such insights will be exploited as we develop our decentralized method in subsequent sections. Fig. 1(c) shows a tree construction using a distributed (algorithmic) method, where the scheduling length of this method depends on the radius of the interference neighbourhood of each link. Our distributed method as well as the interference neighbourhood will be properly introduced and explained in Section VI.

III. Problem Formulation

In this section, we formulate FTCS as an optimization problem whose objective is to obtain a set of forwarding

[1]A configuration is formally defined in Section II-A.

1536-1276 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2015.2512272, IEEE Transactions on Wireless Communications

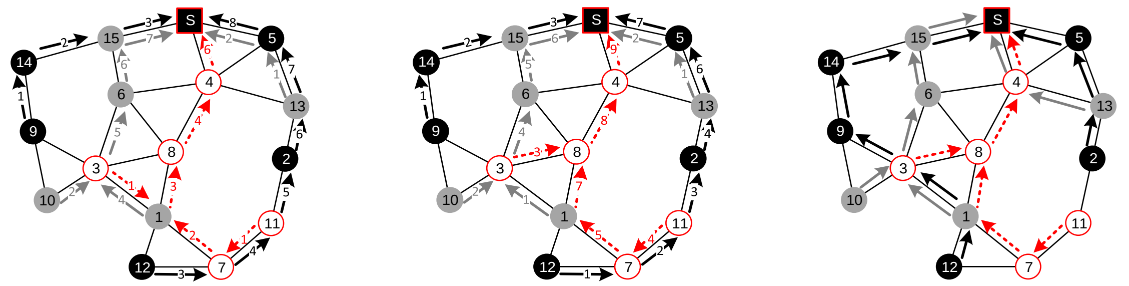

(a) Joint tree construction and scheduling (b) Disjoint tree construction & scheduling (c) Algorithmic tree construction

Fig. 1. FTCS, $m = 3$ ($m = 20\%n$). In the network, the black square S is the sink which intends to gather data from all nodes. Same colour arcs represents aggregation tree for one projection. Each set of interest nodes is illustrated with same colour. The numbers on the arcs represent the time slot when the tree links are active.

TABLE I
NOTATIONS USED IN PROBLEM FORMULATION

Parameters		
V	The set of nodes in the network.	
E	The set of edges in the network.	
n	Total number of nodes.	
m	Total number of projections (trees).	
$\lvert I_t\rvert$	Total number of interest nodes in set I_t.	
P	Node power transmission.	
G_{ij}	Channel gain from transmitter i to receiver j.	
β	SINR threshold.	
η	Background noise.	
S	The set of a large number of time slots sufficient for one round of data gathering (for all transmissions).	
T	The set of m trees required for compressive data gathering.	
ω	Weight of each term in the objective function. ($0 \le \omega < 1$)	
Variables		
f_{ij}^t	$\in \mathbb{N}$	The amount of traffic flow (data traffic load) on link (i,j) in tree t.
x_{ij}^t	$\in \{0,1\}$	Indicating whether link (i,j) is in tree t.
$a_{ij}^{t,s}$	$\in \{0,1\}$	Indicating whether link (i,j) in tree t is active in time slot s.
λ_s	$\in \{0,1\}$	Indicating if at least one link is active at time slot s.

trees which can be scheduled to deliver measurements to the sink in the shortest schedule period to achieve a balance between lower latency delivery and energy efficient gathering. We mathematically formulate the problem as a mixed integer linear program (MILP).

The notations used throughout this section are listed in Table I. The objective of our design is to construct trees which achieve a balance between the number of links needed to gather the measurements (and thus energy expended for data gathering) and the required number of time slots needed to schedule the constructed trees (i.e., gathering latency):

$$Minimize \qquad \omega \sum_{t\in T} \sum_{(i,j)\in E} x_{ij}^t + (1-\omega) \sum_{s\in S} \lambda_s, \tag{3}$$

subject to: (4) - (13), where these constraints will be derived in Sections III-A to III-J.

The first sum in the objective function corresponds to the total number of links in the constructed trees and the second one depicts the scheduling length. The parameter ω ($0 \le \omega < 1$) indicates the weight of each term in the objective. Depending on the task, if one of the terms (whether energy efficiency or time efficiency) is more important than the other, we give more weight for that particular term. Otherwise, we assign equal weight to both terms ($\omega = 0.5$). The following are the constraints for our problem:

A. Traffic Flow conservation constraints

These constraints assert that the total incoming traffic flow (data traffic load) plus the traffic flow originating at a particular node is equal to the total outgoing traffic flow. Let $f_{ij}^t \in \mathbb{N}$ being the data traffic load (number of packets) imposed by certain routing on edge (i,j) or between nodes i and j in tree t. The following constraints, for each tree t, force the set of interest nodes (vector set I_t) which belong to one tree (projection) to have one data flow from each interest node to the sink:

$$\sum_{j:(i,j)\in E} f_{ij}^t - \sum_{j:(j,i)\in E} f_{ji}^t = \begin{cases} -|I_t|, & i = sink; \\ 1, & \forall i \in I_t; \\ 0, & otherwise. \end{cases} \quad \forall t \in T \tag{4}$$

B. Tree link creation constraints

These constraints create forwarding links for a tree. Let $x_{ij}^t \in \{0,1\}$ indicate whether there is a link between nodes i and j in tree t. $x_{ij}^t = 1$, if there is a positive traffic flow from i to j, and zero otherwise. This implies that $f_{ij}^t = 0 \Leftrightarrow x_{ij}^t = 0$ and $f_{ij}^t > 0 \Leftrightarrow x_{ij}^t = 1$ which is achieved by the following inequalities: (note that n (number of nodes) is always greater than any f_{ij}^t)

$$\begin{cases} f_{ij}^t - x_{ij}^t \ge 0 \\ x_{ij}^t - \frac{f_{ij}^t}{n} \ge 0 \end{cases} \qquad \forall (i,j) \in E, \quad t \in T. \tag{5}$$

C. Outgoing link constraints

These constraints assert that each node can have a maximum of one outgoing transmission (link) in each tree to avoid loops. Otherwise, data is not aggregated to a root (sink).

$$\sum_{j:(i,j)\in E} x_{ij}^t \le 1 \qquad \forall i \in V, \quad t \in T. \tag{6}$$

D. Half duplex constraints

The half duplex constraints ensure that a node may not transmit and receive in the same time slot.

$$\sum_{t\in T} a_{ij}^{t,s} + \sum_{t\in T} a_{jk}^{t,s} \le 1 \qquad \forall (i,j) \in E,\ (j,k) \in E,\ s \in S. \tag{7}$$

E. Transmitting constraints

These constraints ensure that a transmitter cannot simultaneously transmit to multiple receivers in the same time slot.

$$\sum_{t\in T}\sum_{j:(i,j)\in E} a_{ij}^{t,s} \le 1 \qquad \forall i \in V, \quad s \in S. \tag{8}$$

F. Receiving constraints

These constraints ensure that a receiver cannot simultaneously receive from multiple senders in the same time slot.

$$\sum_{t\in T}\sum_{i:(i,j)\in E} a_{ij}^{t,s} \le 1 \qquad \forall j \in V, \quad s \in S. \tag{9}$$

G. Link scheduling constraints

These constraints are required to force a link in a tree to be scheduled only once in a time slot.

$$\sum_{s\in S} a_{ij}^{t,s} = x_{ij}^{t} \qquad \forall (i,j) \in E, \quad t \in T. \tag{10}$$

H. Transmission order constraints

These constraints are required to ensure that a node in a tree cannot transmit unless it receives all packets from its children. That is, a link (i,k) in a tree t at time slot s can be scheduled, if all links to its children have been scheduled prior to time slot s (i.e., in time slots between 1 and $s-1$). In other words, $a_{ik}^{t,s}=1$, if $\sum_{\bar{s}=1}^{s-1}\sum_{j:(j,i)\in E} a_{ji}^{t,\bar{s}} \ge \sum_{j:(j,i)\in E} x_{ji}^{t}$.
In LP format, the above condition is written as follows:

$$\sum_{\bar{s}=1}^{s-1}\sum_{j:(j,i)\in E} a_{ji}^{t,\bar{s}} + B(1-a_{ik}^{t,s}) \ge \sum_{j:(j,i)\in E} x_{ji}^{t} \qquad \forall (i,k) \in E, \quad s \in S, \quad t \in T. \tag{11}$$

B is a big constant, which is bigger than the total number of links in any combination of m trees. When $a_{ik}^{t,s}=0$, inequality (11) is always satisfied. But, when $a_{ik}^{t,s}=1$, (11) reduces to $\sum_{\bar{s}=1}^{s-1}\sum_{j:(j,i)\in E} a_{ji}^{t,\bar{s}} \ge \sum_{j:(j,i)\in E} x_{ji}^{t}$ which implies that the summation of all links coming to node i had to be activated at time slots between 1 and $s-1$, otherwise, node i can not transmit (or, link (i,k) can not be active, i.e., $a_{ik}^{t,s} \ne 1$) at the current time slot s.

I. SINR constraints

The following constraints make sure that the SINR for each active link is above the thresholdβ.

$$P.G_{ij} + B_{ij}^{t,s}(1-a_{ij}^{t,s}) \ge \beta(\eta + \sum_{\bar{t}\in T}\sum_{(k,h)\in E;k\ne i} P.G_{kj}.a_{kh}^{\bar{t},s} \qquad \forall (i,j) \in E, \quad s \in S, \quad t \in T. \tag{12}$$

where $B_{ij}^{t,s}$ is a constant and satisfies the following: $B_{ij}^{t,s} \ge \eta + \sum_{t\in T}\sum_{(k,h)\in E;k\ne i} P\,G_{kj}\,a_{kh}^{t,s}$

In (12), if link (i,j) in tree t is active in time slot s (i.e., $a_{ij}^{t,s}=1$), then (12) reduces to expression (1).

J. Finding occupied time slots in a schedule

The following constraints check whether a time slot s has at least one active link or not.

$$\lambda_s \ge a_{ij}^{t,s} \qquad \forall s \in S\ , \quad t \in T\ , \quad (i,j) \in E. \tag{13}$$

Note, after solving the above problem, the time slots which have no active links are removed from the schedule and the remaining time slots form the corresponding solution.

K. NP-hardness

The authors of [5] have shown that the data gathering in WSN under SINR is NP-hard through a reduction from the *max-connections* problem [7]. The max connection problem is to select a maximal set or configuration size under the physical interference model. Our problem however is different from [5] in that we construct multiple forwarding trees (rather than only one) to collect the coded measurements; we also differ in that a node waits for its children's measurements to compress them (with its own) into one packet for upward transmission. This makes the scheduling problem more difficult. The following theorem establishes the hardness of the FTCS problem.

Theorem 1. *The joint problem of forwarding tree construction and scheduling is NP-hard.*

Proof. The FTCS problem has two combined objective terms (constructing m aggregation trees with minimum links, and scheduling these links based on SINR constraint in a shortest time length). Now, according to the weight given to each term, the problem gives different results and hence a different way to prove the NP-hardness.

Stage 1 (giving highest weight to scheduling): Without loss of generality, let us first assume the trees are given. We may prove the minimum link scheduling problem is NP-hard by reducing from the One-Shot Scheduling problem which has been shown to be NP-hard in [6]. The One-Shot Scheduling problem is to pick a subset of weighted links such that the total weight is maximized and the SINR at the receiver of each link is above the threshold β. In other words, attempting to use one slot to its full capacity. Formally, let each link l_i is assigned a weight w_i and r_i indicates the receiver of link l_i. A set $L \subseteq E$ is a solution to One-Shot Scheduling problem if the following conditions hold:

$$L = \max_{\acute{L}\subseteq E}\sum_{l_i\in\acute{L}} w_i, \qquad SINR(r_i) \ge \beta, \forall l_i \in L. \tag{14}$$

It should be noted that in our problem links have equal weights. Therefore, we give a weight of one to each link and the problem of one-shot scheduling becomes of picking a maximum number of links in one slot that satisfies the SINR constraint (w_i=1). The problem of finding the minimum scheduling length among all m data aggregation trees can be decomposed into a series of one-shot scheduling subproblems. In each one-shot scheduling subproblem, an auxiliary graph is constructed (in polynomial time) from a set of links in m aggregation trees that do not have child links for data aggregation. In other words, an edge is added to the auxiliary graph if the corresponding link on any aggregation tree is connected to a leaf node. After resolving the one-shot scheduling subproblem on the auxiliary graph, the scheduled links are removed from the aggregation trees. This step is repeated until no links remain in any tree. Then, the number of iterations is the total number of time slots required for trees scheduling. Therefore, scheduling the problem of finding the minimum scheduling length is NP-hard.

Stage 2 (giving highest weight to tree construction): If we give the highest weight to minimizing the total links in constructing the m aggregation trees, we may prove the problem of constructing each aggregation tree is NP-hard by reducing from the minimum Steiner tree problem which is known to be NP-hard problem [10]. The minimum Steiner tree problem is to connect a set of interest nodes $I \subseteq V$ such that the connected spanning tree has a minimum total distance on its edges. Now, from the minimum Steiner tree problem, if we let one of the nodes in the tree act as a root, the minimum Steiner tree is converted to one tree construction of our problem. Selecting a root (which is the sink node) can clearly be done in polynomial time. We require m such trees for our compressive data gathering. Therefore, the tree construction is also NP-hard. ■

Algorithm 1: Route discovery at node v (**Phase 2**)

```
1  if $root \in N(v)$ then
2      $\pi_v^t = s$;    Set and broadcast $flag_v^t = 1$;
3  else if $b \in N(v)$ AND $b \in I_t$ AND $flag_b^t = 1$ then
4      $\pi_v^t = b$;    Set and broadcast $flag_v^t = 1$;
5  else if $b \in N(v)$ AND $b \in I_t$ AND $h_b^t < h_v^t$ then
6      $\pi_v^t = b$;    Set $flag_v^t = 0$;
7  else if $b \in N(v)$ AND $b \in I_t$ AND $h_b^t = h_v^t$ AND
   successive parents of $b$ reach a node with smaller
   hop-count or $flag = 1$ or non-parent and do not reach $v$
   then
8      $\pi_v^t = b$;    Set $flag_v^t = 0$;
9  else
10     Run BFS from $v$ in a radius equals to $h_v^t - 1$.
11     if interest node(s) in this radius found then
12         Connect $v$ to nearest interest node through
           shortest path.
13         Set $flag_v^t = 0$;
14     else
15         Connect node $v$ through shortest path to the root.
```

IV. Algorithmic solution

To overcome the computational complexity of the FTCS problem, we decompose it into two subproblems, namely the forwarding tree construction and the link scheduling subproblems and present decentralized methods for solving them.

A. Distributed Tree Construction

Our objective is to construct forwarding trees in a decentralized manner. Each forwarding tree will carry a compressed measurement from the network to the sink; our objective is to obtain energy efficient trees which deliver data to the sink with minimal latency.

The compressive data gathering tree construction consists of three phases: 1) disseminating discovery messages; 2) route discovery; 3) search for more efficient routes, to leverage them in the scheduling subproblem. Initially (**Phase 1**), the sink starts by sending a discovery message to its neighbours. Each node, upon receiving the message, will broadcast it to allow other nodes, not close to the sink, to receive the discovery message. This procedure is similar to traversing the network using a breadth-first search (BFS) algorithm [11]. Hence, each node v will learn its shortest path (P_{vs}) to the sink as well as the hop-count (h_v^t) along the path. Further, node v discovers its neighbour set $N(v)$. Node v, upon checking matrix Φ, which is stored in its memory, determines whether node $u \in N(v)$ ($\forall u$) belongs to the set of interest nodes (I_t) of tree t or not. The time complexity for **Phase 1** (similar to BFS) is $O(n)$.

In **Phase 2**, each node v for each tree t (if it is an interest node), after running Algorithm 1, decides its parent on the uplink path to the sink. For each interest node, we assign an attribute to designate its parent interest node (π_v^t) (note, a parent interest node could be a neighbour of v or can be reached through other relay nodes) and a decision flag ($flag_v^t$) to indicate whether the parent interest node of v is fixed. Lines (1-2) show that every interest node which is a neighbour of the root selects the root as its parent node and sets and distributes its decision flag $flag_v^t = 1$. Now, if interest node v (Lines (3-4)) is not a neighbour of the root, but has an interest node neighbour b with $flag_b^t = 1$, then v selects b as its parent interest node and commits its decision ($flag_v^t = 1$). In the case where none of the neighbouring interest nodes (b) of v has its decision flag set (i.e., $flag_b^t = 0$), v will select the neighbouring interest node with the smaller hop-count to the sink as its parent interest node (Lines 5-6). Now, when only interest node neighbours with equal hop-count to the sink as v can be found (Lines 7-8), v selects the one (b) whose successive parents reach an interest node with smaller hop-count or decision flag $flag = 1$ or no parent node, and does not reach node v (to avoid loops). If none of the above conditions is satisfied, v runs a BFS to explore its neighbourhood of radius $h_v^t - 1$ in search for an interest node b with smaller hop-count to the sink; otherwise, it searches for an interest node whose decision flag $flag_b^t = 1$ (Lines 10-13). Node v avoids selecting interest nodes b whose $\pi_b^t = v$ to avoid loops. Finally, if no interest node is found, v connects itself directly through a shortest path to the sink (this path is known from the discovery phase). Node v will repeat the route discovery (Algorithm 1) if it receives a notification message from its neighbours indicating that there is a change in the network, e.g., change in a decision flag, or following a node or link failure due to mobility or channel impairments occurring in the network triggering route maintenance. The time complexity for **Phase 2** is $O(1)$ in the best case (when a node chooses a neighbour node) and $O(\gamma)$ in the worst case (when node does not have a neighbour interest node), where γ is the number of nodes around node v and within a radius $h_v^t - 1$. Note that nodes in the distributed approach simultaneously execute the algorithm.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2015.2512272, IEEE Transactions on Wireless Communications

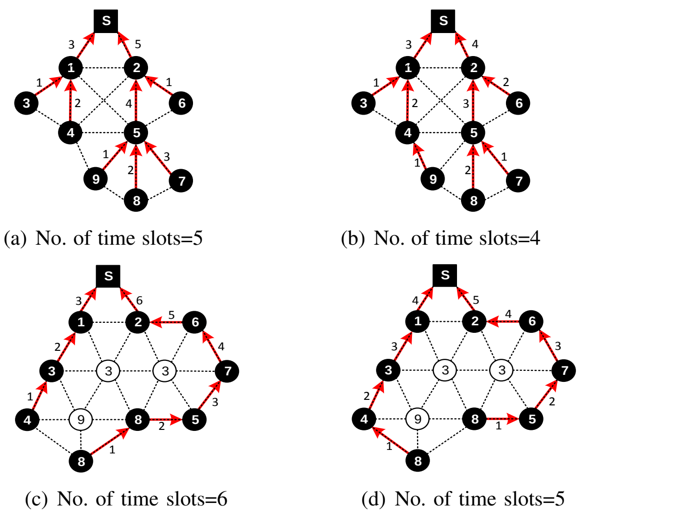

(a) No. of time slots=5 (b) No. of time slots=4 (c) No. of time slots=6 (d) No. of time slots=5

Fig. 2. An example of balancing the node degree and minimizing the height of a tree. In the network, black nodes are interest nodes. The directed arcs denote the links on the data aggregation tree. The active time slot for each arc is shown next to it.

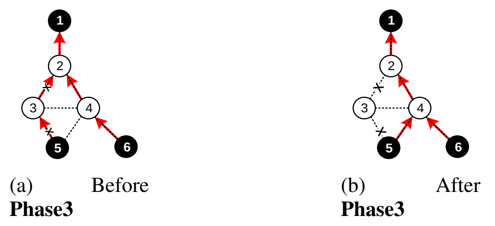

(a) Before **Phase3** (b) After **Phase3**

Fig. 3. An example of removing successive links in a tree. The arcs with X sign denote the removed tree links.

Algorithm 2: Tree refinement at node v (**Phase 3**)

```
1  Remove all the successive tree links from node $v$ to first
   interest node or node that has more than one child in tree
   $t$. Let $b$ be the found node.
2  $R \leftarrow$ Total number of removed links.
3  $Best_{candidate} \leftarrow b$.
4  Run BFS algorithm from $v$ in a radius equals to $R$.
5  if node(s) other than $b$ on disconnected main tree $t$ is
   found in this radius then
6      $Candidates \leftarrow$ put the nearest candidate nodes into
       the list.
7      $Best_{weight}$ = Infinity.
8      for each node $g$ in the $Candidates$ list do
9          $H_g \leftarrow$ Hop-count from $g$ to the sink.
10         $D_g \leftarrow$ Degree of node $g$.
11         if $(D_g + H_g) < Best_{weight}$ then
12             $Best_{weight} = D_g + H_g$.
13             $Best_{candidate} \leftarrow g$.
14 Connect node $v$ to $Best_{candidate}$ in shortest-path.
```

Tree Construction Refinement:

We motivate our refinement phase through an illustrative example shown in Fig. 2(a)-2(b). The intuition for refining the tree selection is that the forwarding trees should have fewer links for energy efficiency and should be scheduled in a shorter time period for latency efficiency. For instance, node 9 may select either node 4 or node 5 as its parent node. Either selection will result in a forwarding tree with same number of links, however, the trees corresponding to the two selections will differ in their data collection latency, obtained from the scheduling subproblem (going through node 5 requires a total of 5 times slots, and through node 4 only 4 time slots). Clearly, if a parent in a tree has a higher node degree, with multiple incoming transmissions, then those transmissions will be scheduled sequentially, and therefore this should be avoided. Clearly, this suggests a thinner but a larger tree height. The larger tree height however may in turn suggest longer schedule period; this is because a parent node along a path towards the sink will have to wait until all downstream measurements are collected before it forwards its own measurement. Recall, measurements have to be compressed, to reduce the number of transmissions in the network. This is depicted in Fig. 2(c)-2(d), where selecting a subtree with larger height increases the scheduling period, and thus collection latency. Motivated by these observations, our tree construction should be refined to yield more efficient forwarding trees, and this is elaborated in **Phase 3**.

In **Phase 3**, each node v checks whether it is among the interest nodes in set I_t. If yes, node v runs Algorithm 2 searching for a more efficient route or a parent that potentially can reduce the scheduling length as discussed above. Algorithm 2 removes all the successive tree links from node v to a node that is either an interest node or has more than one child (an example is illustrated in Fig. 3(a); for node 5, the path shown by arrows with 'x' from node 5 to node 2 is removed (note that node 2 has two children)). Let b represent the node that has more than one child. Let R be the total number of removed links. In this step, an interest node v and its descendants are disconnected from the main tree t (e.g., node 5 in Fig. 3(a) which has been disconnected from the tree). Next, to discover an alternative path to connect v to the main tree t, v searches in a radius equals to R, using Breath-First-Search (BFS), for a node(s) in tree t (if any) that can improve the schedule length and reduce the number of transmissions (e.g., in Fig. 3(b), node 5, which has been disconnected, can connect to the tree through node 4; hence, the overall number of transmissions decreases from 5 to 4). To find a better path, the algorithm adds the nearest candidate nodes found on tree t in a radius R into a $Candidates$ list. Furthermore, for each node g in $Candidates$ list, it retrieves the nodal degree D_g and its hop-distance to the sink H_g. This information can be obtained from each node where they have been obtained from **Phase 2**. As discussed earlier, the candidate that minimizes the nodal degree and the height of the subtree will be selected as the new parent (refer to lines 8-13 in Algorithm 2).

Let δ to be the number of nodes within a radius R, it takes $O(\delta)$ to traverse all nodes in radius R using BFS algorithm. Finding best candidate among nodes in the $Candidates$ list takes $O(\rho)$, where ρ is the size of the $Candidates$ list. Therefore, Algorithm 2 takes $O(\delta + \rho)$, where δ is bigger than ρ, since ρ is a subset of δ. Thus, the time complexity for the algorithm is $O(\delta)$.

1536-1276 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2015.2512272, IEEE Transactions on Wireless Communications

B. Distributed Link Scheduling Algorithm

We consider a Time Division Multiple Access (TDMA)-based access method, and assume time is divided into slots of equal length; we assume each time slot is divided into a scheduling period and a transmission period. A schedule is constructed during the scheduling period where a configuration of links (a configuration is defined earlier) which may be scheduled concurrently is determined. During the transmission period, links in the selected configuration transmit their packets, one packet each, containing their compressed measurements. In this section, we present our decentralized scheduling algorithm, where the objective is for each link to locally schedule its transmission while not violating 1) the order of transmissions and 2) the interference constraints for transmissions to be successful. To achieve this objective, we define for each link an interference neighbourhood, which is centered around the receiver of the link. We shall determine (and control) the cumulative interference caused by active sensors falling in the interference neighbourhood of a link. Further, all links whose transmitters are inside the interference neighbourhood of a link l will be able to exchange information with the transmitter of l for scheduling purposes.

For each link l of length d_l (e.g., a transmission between a transmitter\child i and a receiver\parent j), an interference neighbourhood with a radius $K_l \times d_l$ around the receiver of link l, and using the interference localization method presented in [12], is constructed. The neighbourhood for each link is constructed such that interference beyond this neighbourhood only has negligible impacts on its received signal [12]. For a transmission to be successful on a link l, the maximum interference that can be tolerated at the receiver of link l is:

$$I_l^{max} \triangleq \frac{Pd_l^{-\alpha}}{\beta} \tag{15}$$

where P is the transmit power, α is a power loss exponent and β is a predetermined SINR threshold required for an acceptable bit error rate. The authors of [12] showed that given a constant ϵ, where $0 < \epsilon < 1$, for a link l to be feasible, the upper bound on the interference coming from the transmitters of active links located outside the interference neighbourhood of link l should not exceed ϵI_l^{max} and the total interference coming from transmissions inside the interference neighbourhood cannot exceed $(1-\epsilon)I_l^{max}$. The radius of the interference neighbourhood ($K_l \times d_l$) certainly depends on the value of ϵ. The smaller the value of ϵ, the larger the interference neighbourhood, and thus the higher the scheduling overhead. The value of ϵ can be used to control the scheduling overhead. In addition, the receiver of each link can estimate the interference power created by the transmitter of each link in the interference neighbourhood using the Radio Interference Detection (RID[2].) [13]. For more details about the interference localization and RID methods, we refer the reader to [12] and [13] respectively. It should be noted that other approaches (e.g., FlashLinQ [14] and ITLinQ [15]) have been shown to perform very well in terms of interference management and can be used for our link scheduling subproblem.

[2]The RID protocol is only used to let the receiver estimates the interference caused by any transmitter. The basic idea of RID is that a transmitter broadcasts a High Power Detection (HD) packet, and immediately follows it with a Normal Power Detection (ND) packet. The HD packet contains the transmitters ID, from which the receiver knows from which transmitter the following ND packet comes. The receiver estimates possible interference caused by the transmitter by sensing the power level of the transmitters ND packet. For more details we refer the reader to [13]

Algorithm 3: Distributed Scheduling Algorithm at link l

```
1  Transmitter of link $l$ broadcasts $SchReq$ to all links in $\Delta_l$.
2  Receiver of Links $k \in L \bigcap \Delta_l$ calculate the interference $I_k^{tem}$ after adding link $l$ temporary to $L$.
3  if any receiver of link $k$ has $I_k^{tem} > (1-\epsilon)I_k^{max}$ then
4      Link $k$ sends an $NotAcc$ message to link $l$.
5  if link $l$ receives at least one $NotAcc$ message then
6      Link $l$ does not add itself to schedule $L$.
7      Link $l$ broadcasts $RemSch$ message.
8      All links $k$ upon receiving $RemSch$ message remove link $l$ from current schedule $L$.
9  else if Link $l$ receives no $NotAcc$ messages then
10     Link $l$ is added to the current schedule $L$.
11     Link $l$ broadcasts $AccSch$ message.
12     All links $k$ upon receiving $AccSch$ message update their schedule $L$ by adding link $l$ to $L$.
```

We now propose our distributed scheduling algorithm. Let Δ_l be the set of all links k such that the transmitter of link l is in their interference neighbourhood. Let L be the set of links for the current schedule; at the beginning of each time slot, L is empty. At a high level, links to leaf nodes or links whose children do not have data to transmit will go into a ready state (since they do not have to wait for any downstream data); transmitters of such links broadcast their priority information to all nodes in their interference neighbourhood Δ_l. The priority of each node is estimated based on two criteria: (1) its parent nodal degree and (2) its hop-count to the root (this information is obtained from the tree construction phase). For instance, the priority of a node can be quantified by combining (1) and (2). A node with bigger parent nodal degree and larger hop-count to the sink assigns itself a higher priority. The tie can be broken by the transmitter's node ID (node with bigger ID has higher priority). The priority information of a link l is disseminated to all links (transmitters) within the interference neighbourhood of l. Now, each link l in ready state which has the highest priority among all links (in ready state) in its interference neighbourhood, if its cumulative interference I_l is not exceeding $(1-\epsilon)I_l^{max}$ and its receiver has not already been scheduled for any other link, can simultaneously run Alg. 3 to add itself to the current schedule L. This process continues until no more ready state links can be added to the current schedule L. For the next time slot, new links will be added to the ready state if their predecessor links have been scheduled in the previous time slots. Accordingly, the above procedure will be repeated until no more links are left unscheduled.

V. Performance Analysis

In this section, we prove the correctness of our algorithmic method and analyze its efficiency by giving the approximation

1536-1276 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2015.2512272, IEEE Transactions on Wireless Communications

ratio of the algorithmic tree construction to the optimal one and analyze the performance bounds of the link scheduling algorithm with respect to the aggregation latency.

A. Correctness

Our distributed method, as discussed above, consists of two (tree construction and link scheduling) parts, where the former part has three phases. We prove the correctness of each part or/and phase using the following lemmas.

Lemma 2. *(Correctness of phase 1). The sink disseminates the discovery message, and all the nodes in the network receive it and hence update their information.*

Proof. Sensor nodes in the network are connected and thus there is at least one path from each node to the sink. If nodes upon receiving the discovery message, broadcast it, this guarantees that all the nodes will receive this discovery message, and by updating the hop-counter, nodes learn their distances to the sink, as well as the number of neighbors, since they receive one message from each neighbor. It should be noted that nodes do not re-broadcast the discovery message with equal or higher hop-count, and this proves the termination of the discovery message phase. ■

Lemma 3. *(Correctness of Phase 2). In Algorithm 1, each interest node v that belongs to tree t finds its route to nearest interest node parent.*

Proof. In Algorithm 1, a node has to choose an interest node parent with smaller hop-count (nearer to the sink) or a parent with a decision flag equals one (i.e.; $flag^t_{parent} = 1$). When the node chooses a parent with $flag^t_{parent} = 1$, this guarantees that the route will reach the sink (a node can set its decision flag equal one if its ancestors reach the sink); otherwise, the node will select the parent with smaller hop-count to the sink. The selected parent will repeat the same procedure until the route to the sink is discovered. ■

Lemma 4. *(Correctness of Phase 3). A node in Algorithm 2 can enhance the forwarding tree by finding a more efficient route, if any.*

Proof. After removing the successive tree links from node v and disconnecting it from the forwarding tree t, Algorithm 2 examines all the paths to nearest node(s) in tree t and finally chooses the best one and connects node v to the tree. Hence, reconnecting the disconnected node to the tree ensures the termination of the algorithm. ■

Lemma 5. *(Correctness of Link scheduling). The distributed link scheduling algorithm in Section IV-B can correctly schedule the links in all the m trees under the interference model.*

Proof. Algorithm 3 guarantees that each link in the ready state which has the highest priority among others and its cumulative interference does not exceed the maximum interference which can be tolerated, will be added to current scheduling list and removed from the ready state, and hence will be scheduled once. At the end of each round, the ready state will be updated and links that have not been assigned a time slot remain for future rounds. Finally, all the links will be added to the schedule list and the algorithm terminates. ■

B. Worst Case performance of the tree construction algorithm

In this section, we show that the worst case cost-ratio of our tree construction algorithm to optimal solution will never exceed 2 (i.e.; 2-approximation). The approximation bound of our algorithm follows the results obtained in [16] for the approximation solution of the Steiner tree problem. The Steiner tree problem spans a subset of nodes in a graph (i.e.; $I \subseteq V$) such that the spanning tree obtained from connected interest nodes set I has a minimum cost.

Lemma 6. *[16] For an undirected graph $G(V, E)$ and interest nodes set I, the cost-ratio of minimum Steiner tree algorithm to optimal solution is $2(1 - \frac{1}{l}) \leq 2(1 - \frac{1}{I})$, where l is the number of leaves in the optimal tree.*

Lemma 7. *Our tree construction algorithm is a polynomial time 2-approximation algorithm.*

Proof. We start by noting that our problem for constructing each aggregation tree is similar to the Steiner tree problem in that we connect all interest nodes set $I \subseteq V$ and the sink together such that the constructed spanning tree has a minimum cost on its edges. Based on Theorem 6, the worst case performance ratio of our tree construction algorithm to optimal solution will not be worse than $2(1 - \frac{1}{l})$. It should be noted that the sink is not considered as a leaf node in a tree and hence the cost-ratio will not be worse than $2(1-\frac{1}{I})$. Recall that from Section II-B, the total number of interest nodes for each projection is $\lceil \frac{n}{m} \rceil$. Therefore, the performance ratio of our tree construction method in the worst case is $2(1 - \frac{1}{\lceil \frac{n}{m} \rceil})$. For example, if $n = 100$ and $m = 20$, the upper bound performance of our algorithmic method is $2(1 - \frac{1}{5})$, which is $\frac{8}{5}$-approximation, and when $n = 100$ and $m = 10$, our method is $\frac{9}{5}$-approximation. In compressive data gathering, since the number of projections m is always less than the total number of nodes n and $m \geq \log n$, the fraction ratio of the approximation solution will always be less than 2. Hence, the performance of our tree construction algorithm is better than 2-approximation algorithm depending on $\lceil \frac{n}{m} \rceil$. ■

C. Performance bounds of the link scheduling algorithm

In this section, we discuss the theoretical lower and upper bounds on the latency for data aggregation on the constructed m forwarding trees.

Lemma 8. *Given a set of m trees T for compressive data gathering, the lower bound on the required time slots to schedule all the links in T is*

$$max(m, D^t_i + H^t_i)(\forall i \in V, t \in T) \tag{16}$$

D^t_i and H^t_i are respectively the nodal degree and hop-count to the sink for node i and tree t.

Proof. In any tree, a parent node (doing data aggregation) has to wait until it receives data from all of its children and then forward the aggregated data to its upper node (if it is not a sink

1536-1276 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

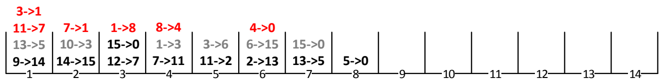
node). Therefore, the minimum number of time slots required for a node i in tree t is D_i^t (i.e., the number of neighbors of node i in tree t). This D_i^t includes the time slot to transmit data from node i to its parent, since its parent has been counted as one of its neighbors in D_i^t.

Now, the minimum time required to send data from a node to the sink is equal to the hop-distance of a node to the sink, which is represented by H_i^t. Therefore, in total a minimum of $D_i^t + H_i^t$ time slots is needed for node i in tree t to send its aggregated data to the sink. If all the transmissions occur in a way that the SINR constraint is satisfied at each receiver, then the possible lower bound on the required number of time slots is $max(D_i^t + H_i^t)(\forall i \in V, t \in T)$. We should note here that the sink can receive only one transmission in each time slot, hence for m trees a minimum of m time slots is required for the sink to receive all the aggregated data from m trees. Therefore, the final lower bound on time slot for CDG is (16). ■

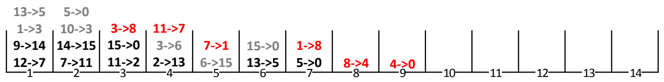
It should be noted that at each time slot, the maximum number of links from m trees which are at the ready state and their SINR is below the threshold β, are going to be scheduled and removed from the trees to let the remaining links to be scheduled in the next rounds. It is possible that in the worst case (because of lack of fulfilling the SINR constraint, common node transmission or receiver among ready state links), no more than one link could be scheduled at each time slot. Intuitively, at least one link can be scheduled at each time slot and thus, the worst case performance of the link scheduling algorithm for m trees under the physical interference model is bounded by the total number of links in all m trees.

VI. Performance Evaluation

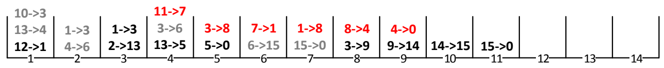
We study the performance of the joint design method under optimal formulation and compare it with the decentralized solution we proposed. We also study the performance of FTCS under optimal tree construction and optimal scheduling, separately. Finally, we compare the performance of our FTCS with LLHC-MWF [5], which does data gathering but not compressive data gathering. Our metrics for comparisons are the number of transmissions and schedule length required to gather data under various network sizes, topologies, and number of projections (for compressive data gathering). For numerical results, we generate arbitrary networks with n nodes where nodes are randomly distributed over a region of 700×700 unit distance, such that the resulting graph is connected. The density (average nodal degree) of the network is tuned by increasing or decreasing the communication range of a node. Further, we randomly assign each node in the network to m sets of interest-nodes where each set contains $\lceil \frac{n}{m} \rceil$ nodes. Note that based on the number of n nodes and m sets, a node might be included in more than one set. We assume all nodes use the same normalized transmit power $P = 1$. Moreover, we assume a path loss exponent $\alpha = 3$ and the SINR threshold for successful transmission $\beta = 2$; we assume the background noise is negligible; we also assume a single transmit rate and hence only one threshold β. We further assume $\omega = 0.5$, giving equal weights to both terms



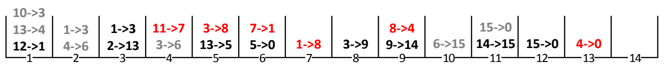
(a) FTCS joint



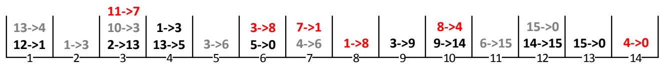
(b) FTCS disjoint



(c) FTCS Decentralized, $\epsilon = 0.3$



(d) FTCS Decentralized, $\epsilon = 0.6$



(e) FTCS Decentralized, $\epsilon = 0.9$

Fig. 4. Link scheduling solution.

in the objective (i.e., energy efficiency and time efficiency are equally important). We use CPLEX to solve our optimization model and JAVA to simulate the operation of our distributed algorithms. We run our program on CPU with Intel Core i7 processor, 3.6 GHz speed, 8 GB memory ram and 64-bit windows operating system.

A. Evaluation on a small network

We start by examining the results obtained by solving the FTCS jointly using the MILP model and compare it with our decentralized solution, using the 15-node network shown in Fig. 1(a). Clearly, both methods construct forwarding trees with same number of links (and thus same number of transmissions to gather the sensory data), however both methods differ in their link scheduling performance as depicted in Fig 4. First, the MILP solution yields an optimal link scheduling (schedule length = 8 time slots), however its solution is centralized. The decentralized solution (see Fig. 1(c)) varies according to the value of ϵ; a smaller ϵ indicates a larger interference neighbourhood and thus larger area to coordinate transmissions and as a result obtain better solutions than larger values of ϵ. However, this better performance comes at the expense of larger scheduling overhead [12]. When $\epsilon = 0.3$, a schedule length of 11 time slots is obtained which is around 27% far from the optimal solution. The MILP however, being a centralized method, exhibits a much higher computational complexity.

B. Centralized Vs. Distributed

Now, we compare the performance of the decentralized solution of FTCS (D-FTCS) with two other disjoint methods, namely, methods that solve the problems of tree construction and scheduling separately using either the centralized optimal model or distributed algorithmic method. The first one is centralized and solves the two subproblems optimally (OTC-OLS)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2015.2512272, IEEE Transactions on Wireless Communications

TABLE II
FTCS PERFORMANCE (NUMBER OF TIME SLOTS, $m = 20\%n$)

#Nodes	Avg. nod. deg.	#Trans.	D-FTCS	DTC-OLS	OTC-OLS
n=10	2.9	10.8	**6**	**5.6**	**5.6**
n=15	3.12	20.6	**9.6**	**9.6**	**9.4**
n=20	3.14	31.4	**14.6**	**12**	**11.4**
n=25	3.46	44.8	**19**	**15.4**	**15.4**
n=30	3.55	63.4	**22.4**	**17.8**	**18.4**
n=35	3.33	82.2	**30.6**	**25.2**	**26**
n=40	5.22	76.6	**29.4**	**21.4**	**22.4**

and the second one solves only the (centralized) scheduling subproblem optimally (DTC-OLS). It should be noted that the link scheduling under interference model is an NP-hard problem, as shown before. The results, averaged over five runs, are shown in Table II. We should recall that the schedule length highlights the gathering latency in the network. It is observed that OTC-OLS and DTC-OLS, being able to provide optimal solutions to the link scheduling subproblem, resulted in shortest schedule length and thus faster collection time for the measurements. It is interesting to note that DTC-OLS for larger network instances resulted in slightly shorter schedules and this is due to the tree construction refinement phase of the distributed algorithm. In OTC-OLS, however, trees are constructed to contain minimum number of edges without any refinement. D-FTCS on the other hand achieved notably a good performance with worst case gap to other solutions not exceeding 27%. In terms of computation complexity, our decentralized algorithm obtained solutions in less than 2 seconds (for a 40 nodes network) whereas OTC-OLS obtained a solution for a 40 nodes network after a day; namely, the tree construction took only few seconds and the link scheduling took 1.5 days. For smaller networks (e.g., 20 nodes), the decentralized method returned the solution in less than one second and OTC-OLS took in the order of minutes. These results confirm that the link scheduling under interference constraints is indeed very complex to solve in a centralized setting. Finally, we should note that all three methods constructed trees with same number of edges, consuming the same number of transmissions. We will further examine and compare the performance of D-FTCS in terms of schedule length and number of transmissions with other method in the literature in subsectionVI-D.

C. Exploring more forwarding trees

At this stage, it should be clear that the scheduling performance depends entirely on the structure of the forwarding trees. In a general graph, more than one forwarding tree with minimum edges may be constructed to gather data from a set of interest nodes to the sink. Therefore, to obtain a more efficient (shorter) schedule, one may first construct all the possible minimum forwarding trees and then solve the scheduling subproblem for all tree combinations, each for a multi-set of interest nodes (or projection), and then choose a combination that gives the best schedule length among others. Let $I_1, I_2, ..., I_m$ represent interest nodes sets for projections $1, 2, ..., m$ respectively. For each set (I_t), we may have different minimum trees (i.e,; $\tau_t = \{t^1, t^2, ...\}$). To find the best scheduling, we have to solve for $\tau_1 \times \tau_2 \times ... \times \tau_m$ combination of trees. Algorithm 4 shows the steps to find all the minimum forwarding trees.

Algorithm 4: Steps to get all minimum forwarding trees

```
1 Construct the optimal forwarding tree. (e.g., (3)-(6)
  without second sum of (3).
    1.1   Add the tree into $MinimumTreesSet$.
    1.2   Let $NLinks$ = number of links in the tree.
  2 Remove links one by one from the optimal tree.
    2.1   Construct tree without that removed link.
    2.2   If the number of links in the obtained tree is
          equal to $NLink$, and the obtained tree is not
          in $MinimumTreesSet$ :
    2.2.1   Put the obtained tree into $MinimumTreesSet$.
    2.2.2   Put the obtained tree into $CheckTreesSet$.
  3 While $CheckTreesSet$ is not empty;
    3.1   Remove one tree from the $CheckTreesSet$.
    3.2   Repeat step 2 for this tree.
```

Recall that a primary objective in a WSN is to minimize the total number of transmissions (links in the forwarding trees) for energy efficiency, and later schedule those trees to obtain a shortest schedule for efficient data gathering latency. If the primary objective is latency, then more trees may be enumerated (step 2.2 in Algorithm 4 can be updated to accept trees with larger size (we add μ to $NLinks$, where, μ indicates the number of edges that is acceptable if the obtained tree has links more than optimal tree)). It might be possible that trees with larger size yield a better schedule length. If we let the value of μ to be large enough (i.e. $\mu \geq$ number of edges in the network), the algorithm will find all possible forwarding trees without a cycle. Let $\overline{\tau_i} = \{\overline{t^1}, \overline{t^2}, ...\}$ be the set of all forwarding trees for each set I_t, the scheduling length is obtained by solving the scheduling subproblem for all $\overline{\tau_1} \times \overline{\tau_2} \times ... \times \overline{\tau_m}$ combinations. Let $S^* = \{\tau_1^*, \tau_2^*, ..., \tau_m^*\}$ indicate the optimal tree combinations yielding optimal schedule, τ_t^* for interest nodes set t (obtained through MILP or exhaustive), and let S^τ and $S^{\overline{\tau}}$ be the best scheduling found for τ and $\overline{\tau}$ respectively. Then, $S^* \leq S^{\overline{\tau}} \leq S^\tau$. Table III shows the results (number of time slots, time complexity and number of constructed forwarding trees combination) for the same instances used in Table II. In this table, the scheduling problem for each trees-combination is solved using the two methods (optimal model and distributed algorithm). The results show that constructing multiple trees for each projection improves the performance of the disjoint methods. However, it significantly increases by computational complexity. For instance, the multiple trees construction with optimal scheduling method (for 20-node network) performed 12% better than OTC-OLS, but obtained the solution after days (res. near an hour) when taking all forwarding tree combinations (res. minimum tree combinations). On the other hand, when solving the scheduling subproblem with distributed algorithm, the multiple trees construction method solved the 20-node network in minutes

1536-1276 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2015.2512272, IEEE Transactions on Wireless Communications

TABLE III
FTCS PERFORMANCE USING MULTIPLE FORWARDING TREES COMBINATIONS ($m = 20\%n$, TIME IS SHOWN BY H:M:S)

#Nodes	Optimal (MILP)		Optimal Scheduling Subproblem				Algorithmic Sheduling Subproblem					
			All-Trees		Min-Trees		All-Trees			Min-Trees		
	#Slots	Time	#Slots	Time	#Slots	Time	#Tran.	#Slots	Time	#Tran.	# Slots	Time
n=10	**5.6**	0:00:23	**5.6**	0:00:59	**5.6**	0:00:24	10.8	**5.6**	0:00:27	10.8	**5.6**	0:00:14
			44.80 tree comb.		16.00 tree comb.		44.80 tree comb.			16.00 tree comb.		
n=15	**7.8**	13:50:37	**7.8**	0:29:35	**7.8**	0:03:16	21.2	**9**	0:01:13	20.6	**9.4**	0:00:38
			427.40 tree comb.		74.2 tree comb.		427.40 tree comb.			74.20 tree comb.		
n=20	Exp. 10	Out of Memory	**10**	61:40:26	**10**	0:40:01	31.4	**12.6**	0:03:07	31	**13**	0:00:51
			3468.40 tree comb.		53.00 tree comb.		3,468.40 tree comb.			53.00 tree comb.		
n=25	–		–		–		44.8	**16**	0:15:08	44.4	**16.4**	0:02:21
							29,425.60 tree comb.			2,034.00 tree comb.		
n=30	–		–		–		–			62.8	**20.8**	0:05:14
							239,831.20 tree comb.			5,301.00 tree comb.		

(much faster than when solving the scheduling subproblem optimally), whereas D-FTCS took less than a second with a worst gap scheduling performance not exceeding 12% (but, equal number of transmissions in case of Min-Trees). It should be noted that the computational complexity of all the multiple trees construction methods grow exponentially with the size of the network, whereas D-FTCS is scalable for very large networks due to the fact that each node in a network can do the tree construction and scheduling locally.

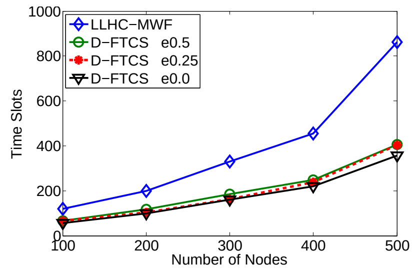
D. Our distributed method Vs. [5]

Next, we examine and compare the performance of our distributed solution for FTCS (D-FTCS) with the centralized data gathering algorithm (LLHC-MWF) presented in [5] in terms of schedule length and number of transmissions required to complete one round of data gathering. It should be noted however that LLHC-MWF algorithm does not use compressive data gathering; it constructs only one tree for data gathering, where each node in the network chooses a parent node that minimizes the maximum subtree size and introduces a new link that is compatible with most links in the constructed tree. Figs. 5(a,b,d,e) depict the results of comparison between D-FTCS (with ϵ=0.0, ϵ=0.25, ϵ=0.5) and LLHC-MWF under different network topologies (sparse and dense) and varying number of sensor nodes (100 to 500) with communication radius ranges from 45 to 100 units for sparse and 75 to 150 units for dense networks, and different number of projections (m=10$\%n$ and m=20$\%n$) with an average of ten runs. As shown in the figures, our D-FTCS (with any value of ϵ) outperforms LLHC-MWF and achieves much shorter schedule lengths (thus lower collection latency). For instance, when $n = 500$ and $\epsilon = 0.5$, D-FTCS in the sparse network performs 25$\%$ and in the dense network performs 21$\%$ better than LLHC-MWF. It should be noted here that such gains are attributed to compressive data gathering, a feature lacking in the LLHC-MWF method. LLHC-MWF on the other hand constructs only one tree and is oblivious to the order of transmissions when performing link scheduling; in other words, in LLHC-MWF, a parent node does not need to wait for its children's measurements since it is not performing any compression, thus its scheduling is more flexible. Nonetheless, our D-FTCS outperformed LLHC-MWF. It is also notable that D-FTCS performs much better than LLHC-MWF when the number of projections is smaller (around 15$\%$ to 25$\%$ when $m = 20\%n$, and 41$\%$ to 52$\%$ when $m = 10\%n$ for different network sizes). With fewer projections, fewer forwarding trees are constructed, and when constructed efficiently, they result in much shorter schedule. Moreover, the curves in the figures show that the performance of D-FTCS over LLHC-MWF in sparse networks increases steeper than dense networks, specially in large networks. The reason goes for the advantage of compressive data gathering, where in the sparse networks, because of deficiency of interference, more links can be scheduled in fewer time slots. In addition, the figures confirm that the results of D-FTCS vary according to the value of ϵ as explained in Section VI-A, where a smaller ϵ achieves shorter schedule length.

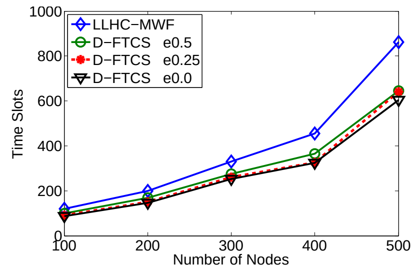
Finally, we consider a network of 200 nodes and we compare the performance of D-FTCS with LLHC-MWF [5] as we vary the number of projections (m) used for FTCS. The results (schedule length and number of transmissions) are shown in Fig. 5(c). The number of transmissions and time slots for different number of projections (m) in LLHC-MWF are both uniform, since this method does not rely on compressive data gathering technique and thus is not affected by different number of projection. The figure shows that when the number of projections is small, D-FTCS substantially outperforms LLHC-MWF both in terms of number of transmissions and schedule length. For instance, when m=5$\%n$ and 10$\%n$, with D-FTCS, few trees are constructed to collect the data from the network (respectively 10 and 20 trees), and owing to compressive data gathering, much fewer transmissions are needed to collect the data (resp. 58$\%$ and 45$\%$ less transmissions), where such transmissions can be scheduled effectively in a very short period of time (resp. performs 67$\%$ and 50$\%$ better). The schedule length is either smaller than half or close to half that of LLHC-MWF. However, as the number of projections increases, then more forwarding trees are constructed and hence more transmissions will be needed. Accordingly, the length of schedule as well as number of transmissions start to increase. As Fig. 5(c) shows, when m=40$\%n$ or bigger, our algorithm performs slightly worse than LLHC-MWF. Alternatively, if the number of projections is kept smaller, then D-FTCS outperforms substantially the performance of

1536-1276 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

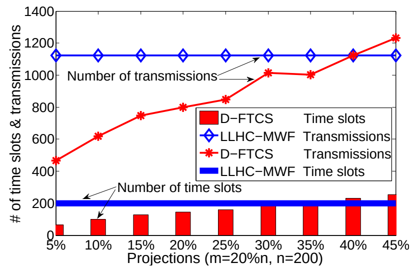
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2015.2512272, IEEE Transactions on Wireless Communications



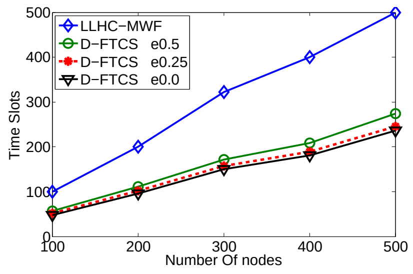
(a) # slots in sparse network, m=10%n



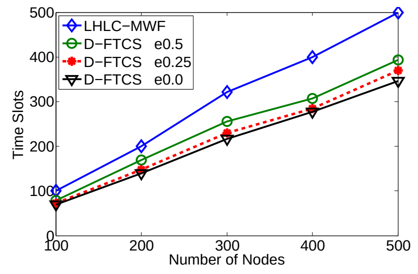
(b) # slots in sparse network, m=20%n



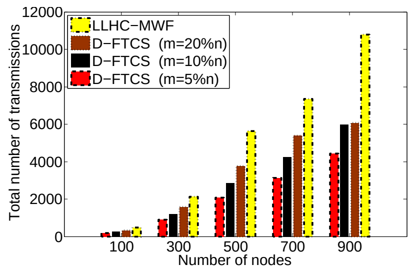
(c) # slots and # transmissions



(d) # slots in dense network, $m = 10\%n$



(e) # slots in dense network, $m = 20\%n$



(f) # of transmissions

Fig. 5. FTCS vs. LLHC-MWF [5]

LLHC-MWF, as depicted in Fig. 5(f), for varying network sizes. For example, when m=20%n, FTCS achieves gains that vary between 29% and 44% over LLHC-MWF.

VII. Related Work

In our previous works [9,17,18], we have studied the problem of sparse random projections for compressive data gathering, however, we did not emphasize on link scheduling. A number of studies in the literature considered the scheduling problem in conjunction with data gathering in wireless sensor networks. Among them, [19] proposed asynchronous distributed data collection using CSMA-based MAC mechanism. The authors in [20] mathematically formulated the problem of joint long-lifetime and minimum latency data collection and aggregation scheduling and proposed an approximation algorithm. The trade-off between energy consumption and time latency was studied in [21]. In [22], the authors presented a distributed implementation for data collection to let each node calculate its duty-cycle locally by giving priority to sub-trees that have bigger size (they assumed the tree is given). The distributed data aggregation scheduling presented in [23] considers interference only from one-hop node. A novel cluster-based TDMA-based MAC protocol for energy-efficient data transmission has been proposed by [24]. In their protocol, for each cluster, a node with higher remaining energy level acts as a cluster head and assigns time slots to all nodes in its cell based on their needs. The authors in [25] and [26] studied the aggregation rate under interference constraint, where [25] tried to maximize the aggregated information at the sink under deadline constraint and [26] tried to minimize the sum delay of sensed data. In [27], the authors investigated the capacity and delay analysis for compressive data gathering under the protocol interference model.

Link scheduling under physical interference model has received increased attention due to its realistic abstraction (e.g.; [28]–[31]). The problem of link scheduling in WSN under the physical interference model was proved to be NP-hard in [6]. In all of these works [28]–[31] the network is partitioned into equal cells and the cells are assigned with colors for concurrent scheduling. In [28] and [29] the aggregation scheduling is done in levels; first aggregate data from nodes in each small area, and then further aggregate data in a larger area by collecting from those small ones. This process is repeated until the entire network as the largest area is covered. [28] constructs the tree and then does the data scheduling in uplink manner, whereas, [29] features joint tree construction and link scheduling by assigning a nearest node to the sink as a cell head. The data collection/aggregation scheme in [30] and [31] is scheduled in two phases, where in each phase the data collection/aggregation is done in one direction (whether horizontally or vertically to next cell). The authors of [31] combined the CDG technique with pipeline technology and came up with more efficient network capacity. In [12], the authors proposed a novel technique under interference localization that allowed them to do scheduling in a decentralized manner. To the best of our knowledge, no other work studied the problem of compressive data gathering under physical interference constraints in a decentralized manner.

VIII. Conclusion

In this paper, we considered the problem of compressive data gathering (CDG) and scheduling in wireless sensor networks under the physical interference model. We formulated the problem of joint forwarding tree construction and link scheduling mathematically with the objective of achieving energy efficient data gathering with minimal collection latency. We highlighted the complexity of the problem, and then we presented our decentralized algorithm for solving it. Our decentralized approach decouples the problem into two subproblems; namely, the tree construction subproblem and the link scheduling subproblem. Our decentralized tree construction is amended with refinements to help the link

1536-1276 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2015.2512272, IEEE Transactions on Wireless Communications

scheduling achieve better scheduling and thus collection latency. Our scheduling subproblem is resolved in a distributed fashion, through interference localization and coordination among links to control the level of interference. Our distributed method showcased the benefits of compressive data gathering in collecting measurements and has been shown to be scalable with outstanding performance in terms of energy efficiency (number of transmissions) and gathering latency (time to gather data from sensors).

References

[1] C. Luo, F. Wu, J. Sun, and C. W. Chen, "Compressive data gathering for large-scale wireless sensor networks," in *Proceedings of the 15th annual international conference on Mobile computing and networking*. ACM, 2009, pp. 145–156.

[2] D. L. Donoho, "Compressed sensing," *Information Theory, IEEE Transactions on*, vol. 52, no. 4, pp. 1289–1306, 2006.

[3] W. Wang, M. Garofalakis, and K. Ramchandran, "Distributed sparse random projections for refinable approximation," in *Proceedings of the 6th international conference on Information processing in sensor networks*. ACM, 2007, pp. 331–339.

[4] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, 2008.

[5] D. Gong and Y. Yang, "Low-latency SINR-based data gathering in wireless sensor networks," in *INFOCOM*. IEEE, 2013.

[6] O. Goussevskaia, Y. A. Oswald, and R. Wattenhofer, "Complexity in geometric SINR," in *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2007, pp. 100–109.

[7] M. Andrews and M. Dinitz, "Maximizing capacity in arbitrary wireless networks in the SINR model: Complexity and game theory," in *INFOCOM 2009, IEEE*. IEEE, 2009, pp. 1332–1340.

[8] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *Information Theory, IEEE Transactions on*, vol. 46, 2000.

[9] D. Ebrahimi and C. Assi, "Compressive data gathering using random projection for energy efficient wireless sensor networks," *Ad Hoc Networks*, vol. 16, pp. 105–119, 2014.

[10] F. K. Hwang, D. S. Richards, and P. Winter, *The Steiner tree problem*. Elsevier, 1992.

[11] T. H. Cormen, *Introduction to algorithms*. MIT press, 2009.

[12] L. B. Le, E. Modiano, C. Joo, and N. B. Shroff, "Longest-queue-first scheduling under SINR interference model," in *Proceedings of the eleventh ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2010.

[13] G. Zhou, T. He, J. A. Stankovic, and T. Abdelzaher, "RID: radio interference detection in wireless sensor networks," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies.* IEEE, 2005.

[14] X. Wu, S. Tavildar, S. Shakkottai, T. Richardson, J. Li, R. Laroia, and A. Jovicic, "FlashLinQ: A synchronous distributed scheduler for peer-to-peer ad hoc networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 21, no. 4, 2013.

[15] N. Naderializadeh and A. S. Avestimehr, "ITLinQ: A new approach for spectrum sharing in device-to-device communication systems," *Selected Areas in Communications, IEEE Journal on*, vol. 32, no. 6, pp. 1139–1151, 2014.

[16] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for steiner trees," *Acta informatica*, vol. 15, no. 2, 1981.

[17] D. Ebrahimi and C. Assi, "Optimal and efficient algorithms for projection-based compressive data gathering," *Communications Letters, IEEE*, vol. 17, no. 8, pp. 1572–1575, 2013.

[18] ——, "A distributed method for compressive data gathering in wireless sensor networks," *Communications Letters, IEEE*, vol. 18, no. 4, pp. 624–627, 2014.

[19] S. Ji and Z. Cai, "Distributed data collection in large-scale asynchronous wireless sensor networks under the generalized physical interference model," *IEEE/ACM Transactions on Networking (ToN)*, vol. 21, no. 4, pp. 1270–1283, 2013.

[20] Z. Chen, G. Yang, L. Chen, and J. Wang, "An algorithm for data aggregation scheduling with long-lifetime and low-latency in wireless sensor networks," *International Journal of Future Generation Communication and Networking*, vol. 5, 2012.

[21] Y. Yu, B. Krishnamachari, and V. Prasanna, "Energy-latency tradeoffs for data gathering in wireless sensor networks," in *INFOCOM 2004.Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*. IEEE,2004.

[22] W.-Z. Song, F. Yuan, R. LaHusen, and B. Shirazi, "Time-optimum packet scheduling for many-to-one routing in wireless sensor networks," *The International Journal of Parallel, Emergent and Distributed Systems*, vol. 22, no. 5, 2007.

[23] B. Yu, J. Li, and Y. Li, "Distributed data aggregation scheduling in wireless sensor networks," in *IEEE INFOCOM*, 2009.

[24] T.-H. Hsu and P.-Y. Yen, "Adaptive time division multiple access-based medium access control protocol for energy conserving and data transmission in wireless sensor networks," *Communications, IET*, vol. 5, no. 18, 2011.

[25] S. Hariharan and N. B. Shroff, "Maximizing aggregated information in sensor networks under deadline constraints," *Automatic Control, IEEE Transactions on*, vol. 56, no. 10, pp. 2369–2380, 2011.

[26] C. Joo, J.-G. Choi, and N. B. Shroff, "Delay performance of scheduling with data aggregation in wireless sensor networks," in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–9.

[27] H. Zheng, S. Xiao, X. Wang, X. Tian, and M. Guizani, "Capacity and delay analysis for data gathering with compressive sensing in wireless sensor networks," *Wireless Communications, IEEE Transactions on*, vol. 12, no. 2, pp. 917–927, 2013.

[28] X. Xu, X.-Y. Li, and M. Song, "Efficient aggregation scheduling in multihop wireless sensor networks with SINR constraints," *Mobile Computing, IEEE Transactions on*, vol. 12, no. 12, pp. 2518–2528, 2013.

[29] H. Li, Q. S. Hua, C. Wu, and F. C. M. Lau, "Minimum-latency aggregation scheduling in wireless sensor networks under physical interference model," in *Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems*. ACM, 2010, pp. 360–367.

[30] S. Chen and Y. Wang, "Data collection capacity of random-deployed wireless sensor networks under physical models," *Tsinghua Science and Technology*, vol. 17, no. 5, pp. 487–498, 2012.

[31] S. Ji, R. Beyah, and Z. Cai, "Snapshot and continuous data collection in probabilistic wireless sensor networks," *Mobile Computing, IEEE Transactions on*, vol. 13, no. 3, pp. 626–637, 2014.

Dariush Ebrahimi received his B.Sc. from Bangalore University in Computer Science, his Master's in Computer Engineering from Kuwait University and he is currently a Ph.D. Candidate in the Department of Computer Science at Concordia University, Montreal, Canada. His research interests are in the area of wireless networks, geographical information system, database, internet of things, optimization and algorithms.

Chadi Assi received his B.Eng. degree from the Lebanese University, Beirut, Lebanon, in 1997 and his Ph.D. degree from the City University of New York (CUNY) in April 2003. He is currently a full professor with the Concordia Institute for Information Systems Engineering, Concordia University. Before joining Concordia University in August 2003 as an assistant professor, he was a visiting researcher with Nokia Research Center. He received the prestigious Mina Rees Dissertation Award from CUNY in August 2002 for his research on wavelength-division multiplexing optical networks. He is on the Editorial Board of IEEE Communications Surveys and Tutorials, IEEE Transactions on Communications, and IEEE Transactions on Vehicular Technologies. His current research interests are in the areas of network design and optimization, network modelling and network reliability.