Maximum-Quality Tree Construction for Deadline-Constrained Aggregation in WSNs

Bahram Alinia, Mohammad H. Hajiesmaili, Ahmad Khonsari, and Noel Crespi, *Senior Member, IEEE*

***Abstract*— In deadline-constrained wireless sensor networks (WSNs), the quality of aggregation (QoA) is determined by the number of participating nodes in the data aggregation process. The previous studies have attempted to propose optimal scheduling algorithms to obtain the maximum QoA assuming a fixed underlying aggregation tree. However, there exists no prior work to address the issue of constructing optimal aggregation tree in deadline-constraints WSNs. The structure of underlying aggregation tree is important since our analysis demonstrates that the ratio between the maximum achievable QoAs of different trees could be as large as $O(2^D)$, where D is the deadline. This paper casts a combinatorial optimization problem to address the optimal tree construction for deadline-constrained data aggregation in WSNs. While the problem is proved to be NP-hard, we employ the Markov approximation framework and devise two distributed algorithms with different computation overheads to find close-to-optimal solution with bounded approximation gap. To further improve the convergence of the Markov-based algorithms, we devise another initial tree construction algorithm with low-computational complexity. Our experimental results from a set of randomly-generated scenarios demonstrate that the proposed algorithms achieve near optimal performance and appreciably outperform methods that work on a fixed aggregation tree by obtaining better quality of aggregation.**

***Index Terms*— Deadline-constrained wireless sensor networks, tree construction, data aggregation, network combinatorial optimization, Markov approximation.**

I. Introduction

A. Motivation

NOWADAYS, monitoring and tracking applications are intrinsically intertwined with a plethora of wireless sensor networks. Data gathering has been considered as a fundamental operation in such applications. In data gathering, limited battery of sensors emphasizes the need for energy-aware data gathering design. However, packet transmission as the major source of energy depletion turns energy conservation in data gathering into an acute problem [2].

To reduce the energy depletion of the sensors due to excessive packet transmission, *data aggregation* [2]–[6] has been proposed as a promising energy conservation mechanism to eliminate the necessity of redundant transmission. In a typical data aggregation scenario, a *data aggregation tree* is constructed over the underlying WSN topology [5] and some intermediate nodes are solicited to aggregate/fuse the gathered data of different sensors by in-network computation and transmit a single packet to the next hop. In this way, the amount of packet transmission is significantly reduced, and hence the overall energy consumption decreases.

Despite the apparent benefits of data aggregation in reducing overall energy usage, it can impose additional delay since the intermediate nodes in aggregation tree must wait to gather sufficient data from the predecessors and then aggregate and forward it to the next hop. This additional delay might be intolerable in many real-time surveillance applications that are sensitive to the latency of the receiving data [7]. For example, in target tracking application, the detected location of a moving object may exhibit perceptible error with the actual location if data aggregation process takes too long [8]. Thus, the imposed delay of a data aggregation algorithm must be considered in an efficient design so as to respect the deadline of the application.

Some previous researches have considered participation of all sensor nodes in data aggregation and aimed to minimize the aggregation delay as the objective [9], [10]. However, participation of all sensor nodes introduces severe interference and may lead to terminating data aggregation in a time that is beyond the application's tolerable delay even though the goal is to minimize the delay. Consequently, these designs fail to guarantee a maximum application-specific tolerable deadline.

B. Deadline-Constrained Data Aggregation: Scenario and Challenges

As a promising alternative, the idea of deadline-constrained data aggregation has been advocated in the recent studies [5], [11]. The general idea is to incorporate a maximum application-specific tolerable delay, namely *deadline*, as a hard constraint, and try to improve the *Quality of Aggregation* (QoA) by increasing the number of participating sensor nodes in data aggregation without missing the deadline. Subsequently, the problem turns into maximizing QoA, subject to the application-specific deadline constraint [5], [11]. Toward this goal, the following two critical challenges should be

Manuscript received November 21, 2016; revised April 9, 2017; accepted April 12, 2017. Date of publication May 5, 2017; date of current version May 22, 2017. This is an expanded paper from the IEEE SENSORS 2015 Conference. The associate editor coordinating the review of this paper and approving it for publication was Prof. Kiseon Kim. *(Corresponding author: Bahram Alinia.)*

B. Alinia and N. Crespi are with the Department RS2M, Institut Mines-Telecom, Telecom SudParis, 91000 Évry, France (e-mail: bahram.alinia@telecom-sudparis.eu; noel.crespi@mines-telecom.fr).

M. H. Hajiesmaili is with the Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: hajiesmaili@jhu.edu).

A. Khonsari is with the School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran 1417466191, Iran, and also with the School of Computer Science, Institute for Research in Fundamental Sciences, Tehran 193955746, Iran (e-mail: ak@ipm.ir).

Digital Object Identifier 10.1109/JSEN.2017.2701552

addressed appropriately: 1) the scheduling policy, and 2) the structure of aggregation tree.

1) Scheduling Policy: Delay in data aggregation is originated from two sources: (i) waiting time to gather the data of predecessor nodes in data aggregation tree, and (ii) waiting time due to the interference issue which is an inherent challenge in wireless networks. If overall waiting time of a node exceeds a specific value, its data cannot be delivered to the sink before the deadline. Devising an efficient policy that schedules the nodes' waiting time while preventing degradation of the QOA and meeting the deadline constraint of the application is a challenging problem. The previous research has tried to find an efficient scheduling such that QOA is maximized [5]. The details are explained in Section II and Appendix.

2) Structure of Aggregation Tree: The structure of data aggregation tree is another important factor. The number of participant nodes in data aggregation could be further improved by constructing a proper data aggregation tree. Without constructing an appropriate aggregation tree, we may not be able to achieve a desired level of QOA even by designing the scheduling algorithm optimally.

The scheduling problem that takes a given tree as input and finds the maximum QOA for the given tree, has been investigated in previous studies [5], [11], under tree topology and one-hop interference mode. We show that the structure of the underlying aggregation tree plays an important role in QOA. Consequently, the ultimate optimal design cannot be fully achieved without taking this critical issue into account. Motivated by this fact, the goal of this paper is to study the problem of constructing an optimal underlying data aggregation tree in a graph topology and under protocol interference model [12].

C. Summary of Contributions

In this paper, we formulate the problem of maximizing QOA in deadline-constrained WSNs under protocol interference model and aim to develop a tree construction and scheduling algorithm to maximize QOA. However, constructing the optimal aggregation tree over a general topology is a network combinatorial problem which is nontrivial even in centralized manner. This is more problematic when we seek an appropriate solution amenable to distributed realization so as the sensor nodes choose their parents (in data aggregation tree) just using local information. We tackle this problem in single-sink WSNs setting through the following contributions:

- We investigate the impact of data aggregation tree structure on QOA by theoretical analysis and explanatory example. We show that the ratio between the maximum achievable QoAs of two data aggregation trees is $O(2^D)$ in the worst case where D is the aggregation deadline. This observation makes the problem of constructing maximum QOA tree intriguing. Besides, we prove that the problem of optimal tree construction belongs to the class of NP-hard problems.
- After formulating the underlying tree construction problem, we leverage Markov approximation framework [13] as a general framework toward solving combinatorial network problems in distributed fashion. By addressing the unique challenges of our problem, we devise two close-to-optimal algorithms in which the sensor nodes contribute to migrating toward a near optimal tree in an iterative and distributed manner. The highlights are bounded approximation gap, and robustness against the error of global estimation of WSN by local information.
- To further improve QOA and convergence of Markov-based algorithms, we propose an initial tree construction algorithm, called *FastInitTree*, as the initialization step of the main algorithm. The algorithm features low computational complexity and close-to-optimal estimate for the initial tree construction, i.e., the QOA obtained by initial tree constructed by *FastInitTree* is close to the QOA achieved when the main algorithm converges. We analyze the validity of constructed trees when the deadline value is changed and prove that for the cases that the deadline decreases there is no need to executed the algorithms again and construct a new tree.
- Through experiments, we evaluate the performance of the proposed algorithms by comparing them to the optimum and the case with fixed aggregation tree. Obtained results demonstrate that four presented algorithms are near-optimal and greatly increase the QOA compared to the method that merely uses random tree to find optimal scheduling without optimal tree construction.

D. Paper Organization

The rest of this paper is organized as follows. We review the related work in Section II. In Section III, the system model is introduced and by motivating examples and theoretical analysis, the impact of aggregation tree on QOA is investigated. Problem formulation and NP-hardness analysis are explained in Section IV. In Section V, we devise two distributed algorithms for the problem. In Section VI, we explain our tree initialization algorithm. Simulation results are described in Section VII. Finally, concluding remarks and future directions are mentioned in Section VIII.

II. Related Work

A. Minimum Delay and Deadline-Constrained Aggregation

The problem of minimum delay data aggregation has been tackled intensively in the literature. In [14], it is proved that the minimum latency aggregation scheduling problem is NP-hard and a $(\Delta - 1)$-approximation algorithm has been presented where Δ is the maximum node degree in the network. The current best approximation algorithms in [10] and [15] achieve an upper bound of $O(\Delta + R)$ on data aggregation delay where R is the network radius. While most studies consider a protocol interference model, the studies in [9] and [16] assume a physical interference model that is more practical than the former. In [16], a scheduling algorithm for tree-based data aggregation is designed that achieves a constant approximation ratio by bounding the delay at $O(\Delta + R)$. The work is extended in [9] for any arbitrary network topology. A connected dominating set or maximum independent sets are employed in [17] to provide a latency bound of $4R' + 2\Delta - 2$ where R' is inferior network radius.

Within the context of deadline-constrained data aggregation models, the goal is not to minimize the delay as an objective of the problem. Rather, the objective is to maximize the number of sensor nodes participating in aggregation while respecting the application-specific deadline. This type of real-time data aggregation has recently gained attention in several works [5], [8], [11], [18], [19]. In this regard, [5] presented a polynomial time optimal algorithm for the problem under the deadline and one-hop interference constraints. The problem is extended in [8] for a network with unreliable links under an additional constraint on nodes' energy level. In [8], the authors proved that in a network with V nodes, the problem is NP-hard when the maximum node degree of the aggregation tree is Δ. They proposed a polynomial-time exact algorithm when $\Delta = O(\log V)$. In [11], the authors considered the same problem of [5] by taking into account the effect of data redundancy and spatial dispersion of the participants in the quality of final aggregation result and proposed an approximate solution for proved NP-hard problem. In a more general case, [19] tackles the utility maximization problem in deadline constrained data aggregation and collection then provides efficient approximation solutions. A main drawback of the aforementioned studies is that they all have tried to maximize the quality of data aggregation on a given tree and neglect the impact of the data aggregation tree structure.

B. Optimum Aggregation Tree Construction

Several studies have tackled the problem of constructing optimal data aggregation tree [20]–[25] where all have been shown to be NP-hard. The study in [22] considers a sensor network composed of source and non-source nodes. Then, the problem is to construct an aggregation tree such that the minimum number of non-source nodes included. In [23], the problem of maximum lifetime aggregation tree is studied for single sink WSNs. The problem is extended for multi-sink WSNs in [21]. Also, [24] studies the problem in large scale WSNs. The problem of constructing an aggregation tree in order to minimize total energy cost is addressed in [25]. As solution, a constant factor approximation algorithm is proposed. In [20], the problem of constructing a minimum cost aggregation tree under Information Quality (IQ) constraint has been tackled. The authors considered event-detection WSNs and defined IQ as detection accuracy. [26] shows that for the shortest path trees, the problem of building maximum lifetime data aggregation tree can be solved in polynomial time and propose two centralized and distributed algorithms. In this paper, we target the construction of maximum quality aggregation tree under deadline constraint that has been overlooked in the previous studies.

Moreover, while Markov approximation framework has been used in different applications such as in P2P streaming [27] and multimedia networking [28], this work is the first that applies the framework in wireless sensor network. In this way, our solution method is completely different from the previous studies and can be considered as a potential solution for the same category of problems.

TABLE I

SUMMARY OF KEY NOTATIONS

Notation	Definition
$\mathcal{V}$	Set of sensor nodes, $V = \lvert\mathcal{V}\rvert$
$\mathcal{T}(\mathcal{G})$	Set of all spanning trees in graph $\mathcal{G}$
D	Sink deadline
$H^{\psi}(i) \subseteq \mathcal{V}$	The set that consists of node i and all its predecessors (except the sink) in tree ψ
F_i	$F_i = 1$, if node i is a source, $F_i = 0$, otherwise
n_i^{ψ}	$n_i^{\psi} = 1$, if node i in tree ψ is allowed to send data to its parent, $n_i^{\psi} = 0$, otherwise $(\vec{n}^{\psi} = [n_i^{\psi}, i \in \mathcal{V}])$
W_i^{ψ}	Waiting time of *participant* node i in aggregation tree ψ, $(\vec{W}^{\psi} = [W_i^{\psi}, i \in \mathcal{V}])$
$\mathrm{QoA}(\psi, \vec{W}^{\psi}, D)$	**Objective function:** The QoA in tree ψ and deadline D and assigned waiting times $\vec{W}^{\psi}$

III. SYSTEM MODEL AND PROBLEM MOTIVATION

A. WSN System Model

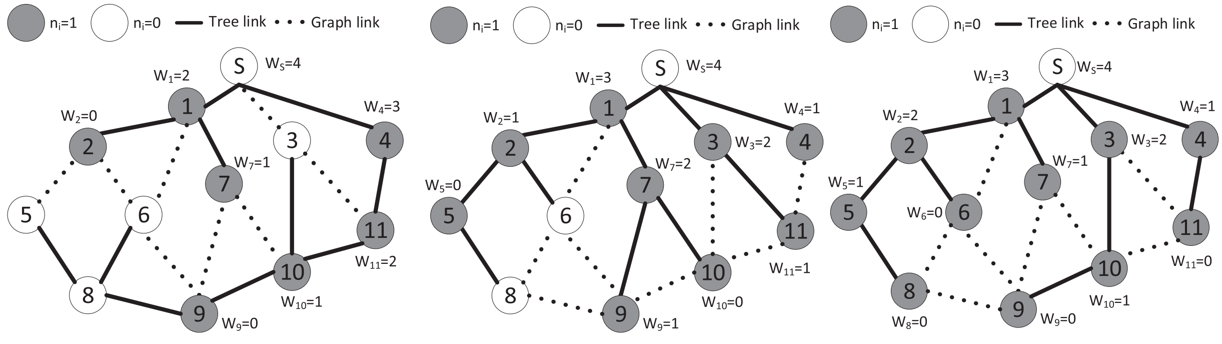
Consider a WSN whose topology is a graph $\mathcal{G} = (\mathcal{V} \cup \{S\}, \xi)$ where S is the sink node, $\mathcal{V}$ is the set of sensor nodes with $|\mathcal{V}| = V$, and ξ is the set of links between sensor nodes. We assume that all nodes have a fixed communication range of R_C and $(i, j) \in \xi$ if nodes i and j are adjacent, i.e., they are in the communication range of each other, i.e., $d(i, j) \leq R_C$. Without loss of generality, we assume that each link has a unit capacity. We suppose that the system is time-slotted and synchronized and a transmission takes exactly one time slot. In deadline-constrained scenario, the data has to be received by the sink by the end of at most D time slots, where the value of D is specified by the deadline requirement of the applications. We adopt the general protocol interference model [12] where at any time slot $t, t = 0, \ldots, D - 1$ transmission over link $m \in \mathcal{E}$ with m_o and m_d as sender and receiver nodes is successful if for each link $l \in \mathcal{L}\backslash\{m\}$ with sender l_o and receiver l_d we have

$$d(l_o, m_d) \geq (1 + \delta) d(m_o, m_d) \text{ and } d(m_o, m_d) \leq R_C, \quad (1)$$

where $\mathcal{L}$ is set of active links at time slot t, δ is a positive constant, and R_C is the communication range of nodes.

Under the described system model, the data aggregation is carried out by running a scheduling algorithm over a constructed spanning tree $\psi \in \mathcal{T}(\mathcal{G})$ on top of the underlying WSN topology where $\mathcal{T}(\mathcal{G})$ is the set of all spanning trees in graph $\mathcal{G}$. The scheduling algorithm outputs a feasible scheduling on the aggregation tree. Specially, the algorithm provides interference-free transmissions according to the specified protocol interference model. The authors in [5] proposed a scheduling algorithm for tree topology under a simplified interference model namely one-hop where transmissions over two links interferes if they have a node in common. In Section V-C we extend this algorithm to work on graph topology and with protocol interference model.

Let $H^{\psi}(i) \subseteq \mathcal{V}$ be the set that consists of node i and all its predecessors (except the sink) in aggregation tree ψ.



(a) (b) (c)

Fig. 1. Impact of aggregation tree structure on the maximum QoA. At each slot $t, t = 0, \dots, D-1$, all nodes having waiting time t send their data in parallel (e.g., in Fig. 1a, nodes $\{2, 9\}$ at $t = 0$, $\{10, 7\}$ at $t = 1$, $\{11, 2\}$ at $t = 3$, and $\{4\}$ at $t = 4$). (a) Long tree: QOA = 7. (b) Random tree: QOA = 9. (c) Optimal tree: QOA = 11.

We consider two types of nodes, source nodes and relay nodes. Source nodes can sense their own data and forward/aggregate the other nodes' data. Relay nodes just forward/aggregate the data of other nodes. To illustrate this, we use binary variable F_i, where $F_i = 1$, if node i is a source and $F_i = 0$, otherwise. Moreover, we define binary variable n_i^{ψ} where $n_i^{\psi} = 1$ indicates that node i in tree ψ is allowed to send data to its parent denoted by P_i and, $\vec{n}^{\psi} = [n_i^{\psi}, i \in \mathcal{V}]$. Indeed, $n_i^{\psi} = 1$ indicates that node i participates in data aggregation. In this case, if $F_i = 1$ then node i is a source participant, otherwise node i participates in data aggregation as a relay node, i.e., it just aggregates the received data from its successors and forwards to its parent.

Let $\mathcal{V}_{\text{leaf}}^{\psi} \subseteq \mathcal{V}$ be the set of all leaf nodes and $\mathcal{V}_{\text{sel-src}}^{\psi} \subseteq \mathcal{V}$ be the set of source nodes *selected* for data aggregation in tree ψ. Indeed, $i \in \mathcal{V}_{\text{sel-src}}^{\psi}$, if i is a source and all of its predecessors are selected for aggregation, i.e., $\mathcal{V}_{\text{sel-src}}^{\psi} = \left\{ i \in \mathcal{V} : F_i = 1 \text{ and } \prod_{j \in H^{\psi}(i)} n_j^{\psi} = 1 \right\}$.

To devise a feasible aggregation scheme, we assign a waiting time of $W_i^{\psi}, 0 \leq W_i^{\psi} \leq D$ time slots to each *participant* node i in aggregation tree ψ and $\vec{W}^{\psi} = [W_i^{\psi}, i \in \mathcal{V}]$. When we run a deterministic scheduling algorithm over aggregation tree ψ with parameter D, $\vec{W}^{\psi}$ denotes the assigned waiting times and $\text{QoA}(\psi, \vec{W}^{\psi}, D)$ determines quality of aggregation which is equal to the number of source participant nodes in data aggregation, i.e.,

$$\text{QoA}(\psi, \vec{W}^{\psi}, D) = |\mathcal{V}_{\text{sel-src}}^{\psi}| = \sum_{i \in \mathcal{V}} F_i \prod_{j \in H^{\psi}(i)} n_j^{\psi}. \quad (2)$$

For notational convenience, we define $\text{QoA}(\psi_i, \vec{W}^{\psi}, D)$ as QoA of sub-tree ψ_i of tree ψ rooted at node i. Hereafter, we use QoA and W_i as a brief notations of $\text{QoA}(\psi, \vec{W}_{\psi}, D)$ and W_i^{ψ} when the corresponding tree and scheduling are obvious, or a specific tree or scheduling is not the matter of concern. The summary of notations are listed in Table I.

B. The Impact of Aggregation Tree

First, note that the optimal aggregation does not follow any particular pattern. For example, *chain-like* long trees are not proper structure for data aggregation trees. The reason is that when sink imposes a deadline D, all nodes with height greater than D cannot participate in data aggregation due to the delay constraint. Consequently, the height of the tree is limited to D and long trees cannot be proper structures. Instead, one might suggest a tree so as the height of the majority of nodes is less than D. However, the waiting time of a node with height h is upper bounded by $D - h$ and hence it can choose at most $D - h$ children of itself as the participants. Hence, the others together with their successors are ignored. Thus, same as the long tree, a *star-like* fat tree may yield a non-optimal QoA. *Generally, an aggregation tree which is neither so long nor so fat is suitable.* Note that the above observations cannot bring significant insights to devise an algorithm to construct the optimal tree. In the next example, we demonstrate that maximum QoA of two aggregation trees of a same network can be different even using optimal scheduling policy. For details on the scheduling policy we refer the reader to Appendix.

Example 1 (The Impact of Aggregation Tree on Maximum Achievable QoA*):* Fig. 1 illustrates the maximum achievable QoA of three different data aggregation trees given a fixed underlying WSN topology. Fig. 1a is an example of long tree. With sink deadline 4, at most one node in height 4 of aggregation tree can participate in data aggregation. That is, just one of the nodes 3 and 9, both with height 4, can participate in the aggregation. Moreover, the set of nodes $\{5, 6, 8\}$ are in a distance greater than D and there is no way to participate them. In other words, this particular long tree structure already has no way to participate at least 4 nodes in aggregation process. Using exhaustive search, it turns out that the maximum QoA of tree in Fig. 1a is 7. Fig. 1b shows a random tree with the maximum QoA of 9. Finally, the optimal data aggregation tree is shown in Fig. 1c where all nodes are participants. The optimal tree in this toy example is obtained by trial and error. We emphasize that finding the optimal aggregation tree is not straightforward even in our tractable topology with only 12 nodes, while in practice the scale of the network is much larger than that of this example.

Theorem 1: For an imposed deadline D where all nodes are source, the maximum values of QoA *in the optimal tree and worst-case tree are $2^D - 1$ and D, respectively.*

Proof: It is proved in [11] that under one-hop interference model, QoA is bounded to 2^D-1 regardless of the aggregation tree structure. Note that the bound is valid also when using protocol interference model since the protocol model covers all interference identified by one-hop model. Therefore, by switching to the protocol model, the QoA cannot be increased. The QoA achieves this bound when the network graph is dense enough where an obvious case is a complete graph (for more details, refer to Section VI). Therefore, we proceed to calculate the upper bound in the worst case. Indeed, the worst case occurs when we construct a chain-like tree with sink as the head of the chain. Observe that for a node i, $|H^{\psi}(i)|$ is equal to the hop distance of i to the sink in aggregation tree ψ. In a chain tree, there is only one possible way of scheduling where each node i having the property $|H^{\psi}(i)| \leq D$ assigned a waiting time of $D - |H^{\psi}(i)|$ and is a participant. There are D such nodes and the maximum QoA of the tree is D. □

The motivating example and Theorem 1 confirm that the structure of aggregation tree plays an important role on the final QoA. In the next section, we formulate the optimal aggregation tree construction as an optimization problem.

IV. Problem Formulation

We formulate the problem of maximizing QoA in a graph topology and under the specified protocol interference model in Section III-A as Tree Construction and Scheduling Problem (TCSP) as below:

$$\begin{aligned}
\mathsf{TCSP}: \ \max_{\psi,\vec{W}^{\psi}} \quad & \text{QoA}(\psi,\vec{W}^{\psi},D)\\
\text{s.t.} \quad & \forall(i,j,k,l) \in \Big\{(i,j,k,l) : (i,j),(k,l)\\
& \quad \in \xi^{\psi}, j = P_i, l = P_k, l \neq j, W_i^{\psi} = W_k^{\psi}\Big\}\\
& d(i,l) \geq (1+\delta)R_C, d(i,j) \leq R_C, \qquad \text{(3a)}\\
& W_i^{\psi} \in \{0,1,\ldots,D-1\}, \quad \forall i \in \mathcal{V}, \qquad \text{(3b)}\\
& W_S^{\psi} = D, \qquad \text{(3c)}\\
& n_i^{\psi} \in \{0,1\}, \quad \forall i \in \mathcal{V}, \qquad \text{(3d)}\\
& \psi \in \mathcal{T}(\mathcal{G}). \qquad \text{(3e)}
\end{aligned}$$

The goal is to maximize QoA by choosing the right aggregation tree ψ and scheduling (assigning waiting times to nodes). Constraint (3a) enforces the protocol interference model to ensure that in any transmission $i \to j$ which is occurring simultaneously with transmission $k \to l$, i.e., $W_i^{\psi} = W_k^{\psi}$, the receiver node l is outside of node i's interference range. Constraints (3b)-(3e) enforce the feasible set of waiting times and spanning tree according to the definitions.

A. NP-Hardness

The problem of finding the optimal tree is hard to solve as the number of trees in the network is extremely large in reality. For example, in a complete network graph with V nodes and a sink, the number of feasible trees is V^{V-2}. We prove that the TCSP is at least as hard as a variant of classical Maximum Coverage Problem (MCP) called Maximum Coverage Problem with Group Budget Constraint (MCPG) which is known to be NP-hard [29].

1) Maximum Coverage Problem: Given a collection of n sets $U = \{S_1, S_2, \ldots, S_n\}$ and a number l, the goal of the MCP is to form set U' by choosing at most l sets from U such that the union of selected sets has the maximum cardinality:

$$\mathsf{MCP}: \ \max_{U'} \left| \bigcup_{S_i \in U'} S_i \right|, \qquad \text{s.t.} \ \ U' \subseteq U, \quad |U'| \leq l.$$

2) Maximum Coverage Problem With Group Budget Constraint: In [29], the MCPG is introduced as a general case of the MCP. In the MCPG, n sets $S_1, \ldots, S_n$ at the MCP are partitioned to L groups $G_1, \ldots, G_L$. The MCPG has two versions namely cost and cardinality versions where the latter is our interest. In the cardinality version of the MCPG, given number l, we should select at most l sets from U such that the cardinality of union of the selected sets is maximized. Moreover, we are permitted to choose at most one set of each group. The MCPG is clearly NP-hard because the MCP which is known to be NP-hard [29] is a special case of the MCPG where each set in U is considered as a group.

$$\begin{aligned}
\mathsf{MCPG}: \ \max_{U'} & \left| \bigcup_{S_i \in U'} S_i \right|\\
\text{s.t.} \ \ & U' \subseteq U, \quad |U'| \leq l,\\
& |U' \cap G_i| \leq 1, \forall i \in \{1,\ldots,L\}.
\end{aligned}$$

The similarity between our tree construction problem and the MCPG is that in both cases the objective is to maximize the cardinality. In the MCPG we can choose at most one set from each group. Similarly, in the TCSP, each node can subscribe (cover) different set of sensor nodes based on its deadline and we are allowed to choose at most one set according to the assigned deadline.

Theorem 2: The TCSP *is NP-hard.*

Proof: To prove, we reduce the MCPG to the TCSP with a polynomial time algorithm. To this end, we construct network graph $\mathcal{G}$ such that the sink is directly connected to L non-source sensor nodes $C_1, \ldots, C_L$ where L is the number of groups in the MCPG. There are V other sensor nodes all considered as source nodes connected to $C_1, \ldots, C_L$ either directly or indirectly where V is equal to the total number of distinct elements in all groups. That is, $V = \sum_{i=1}^{L}\sum_{j=1}^{|G_i|}|g_{i,j}|$ where $|g_{ij}|$ is the cardinality of j^{th} set in group i and $|G_i|$ is the number of sets in group i. Then, we set the sink deadline to $D \geq N$ where N is the total number of sets in L groups, i.e., $N = \sum_{i=1}^{L}|G_i|$. We connect V sensor nodes to $C_1, \ldots, C_L$ and to each other such that if we assign a deadline of $D-((\sum_{k=1}^{i-1}|G_k|)+j-1)$ to the sink's neighbor C_i, j^{th} set of $G_i, 1 \leq j \leq |G_i|$ denotes the maximum cardinality set of the sensor nodes who will participate in data aggregation as the successors of C_i in a sub-tree rooted at this node in aggregation tree. An optimal assignment of deadlines to $C_1, \ldots, C_L$ is equal to select at most one set from each group of the MCPG where this optimal assignment results in maximizing both the number of participants in data aggregation tree as well as the number of covered elements in the MCPG. Therefore, a polynomial time optimal algorithm of the TCSP leads to a polynomial solution of the MCPG which completes the proof. □

V. Markov-Based Approximate Solution

Since the TCSP is NP-hard, it is not possible to devise a computationally-efficient algorithm for the optimal solution even in a centralized manner. As such, we pursue approximate solutions. Among different approximation methods, we leverage Markov approximation framework [13] to propose an efficient near-optimal solution for the problem. Generally, in this framework the goal is to tackle combinatorial optimization problems in distributed manner so as 1) to construct a class of problem-specific Markov chains with a target steady-state distribution and 2) to investigate a particular structure of Markov chain that is amenable to distributed implementation. We first begin with a brief primer of the theoretical approximation framework [13] in the next subsection.

A. Markov Approximation

As indicated in Table I, $\mathcal{T}(\mathcal{G})$ denotes the set of all possible trees (configurations) of the network. For notational convenience, let us define $\Phi_{\psi}^{D} = \mathrm{QoA}(\psi, \vec{W}_{\psi}^{*}, D)$ under constraint set in Equations (3b)-(3e) where $\vec{W}_{\psi}^{*}$ is the optimal scheduling in tree ψ, i.e., when the network relies on aggregation tree $\psi \in \mathcal{T}$ and sink deadline is D, the maximum data aggregation quality is Φ_{ψ}^{D}. Denote p_{ψ} as the fraction of time that data aggregation tree ψ is used to accomplish data aggregation. Using these notations we can rewrite the TCSP as follows:

$$\mathsf{TCSP^{eq}}: \max_{\{p_{\psi}\geq 0, \psi\in\mathcal{T}\}} \sum_{\psi\in\mathcal{T}} p_{\psi}\Phi_{\psi}^{D}, \quad \text{s.t.} \sum_{\psi\in\mathcal{T}} p_{\psi} = 1.$$

Note that the constraints in the TCSP are not appeared in the $\mathsf{TCSP^{eq}}$ because the value Φ_{ψ}^{D} is obtained by respecting the constraints in the TCSP. To derive a closed-form of the optimal solution of the $\mathsf{TCSP^{eq}}$ and to open new design space for finding a distributed algorithm, we formulate the TCSP^{β} as an approximate version of the $\mathsf{TCSP^{eq}}$ using *log-sum-exp* approximation [13]

$$\mathsf{TCSP}^{\beta}: \max_{\{p_{\psi}\geq 0, \psi\in\mathcal{T}\}} \sum_{\psi\in\mathcal{T}} p_{\psi}\Phi_{\psi}^{D} - \frac{1}{\beta}\sum_{\psi\in\mathcal{T}} p_{\psi}\log p_{\psi}$$
$$\text{s.t.} \sum_{\psi\in\mathcal{T}} p_{\psi} = 1,$$

where β is a large enough positive constant that controls the accuracy of the approximation. The TCSP^{β} is an approximate version of the TCSP off by an entropy term $-\frac{1}{\beta}\sum_{\psi\in\mathcal{T}} p_{\psi}\log p_{\psi}$ and it is a convex optimization problem. Hence, by solving KKT conditions its optimal solution is

$$p_{\psi}^{*} = \frac{\exp\left(\beta\Phi_{\psi}^{D}\right)}{\sum_{\psi'\in\mathcal{T}}\exp\left(\beta\Phi_{\psi'}^{D}\right)}, \quad \psi \in \mathcal{T}. \tag{4}$$

Moreover, the optimal value is

$$\widehat{\Phi}_{\psi}^{D} = -\frac{1}{\beta}\log\left(\sum_{\psi\in\mathcal{T}}\exp\left(\beta\Phi_{\psi}^{D}\right)\right). \tag{5}$$

Finally, the approximation gap is characterized as:

$$|\max_{\psi\in\mathcal{T}}\Phi_{\psi}^{D} - \widehat{\Phi}_{\psi}^{D}| \leq \frac{1}{\beta}\log|\mathcal{T}|, \tag{6}$$

where the approximation gap approaches to zero as β approaches to infinity. This means that with larger values of β the approximation model is more accurate.

In the next step, we obtain the solution of the TCSP^{β} by time-sharing among different data aggregation trees according to p_{ψ}^{*} in Equation (4). According to the basic framework, the key is to investigate a well-structured and distributed-friendly Markov chain whose stationary distribution is p_{ψ}^{*}.

B. Markov Chain Design

We design a time-reversible Markov chain with states space $\mathcal{T}$ and the stationary distribution p_{ψ}^{*}. Then, we use this Markov chain structure to hop (migrate) among different states (trees) such that a tree with high QoA has more chances to be visited by Markov random walks. The problem attains its solution when the Markov chain converges to the ideal steady-state distribution.

Given the Markov chain state space, the next step is to construct the transition rate between two states. Let $\psi, \psi' \in \mathcal{T}$ be two states of Markov chain and $q_{\psi,\psi'}$ be the transition rate from ψ to ψ'. Herein, the theoretical framework enriches us by two degrees of freedom. It turns out that the key in designing distributed algorithms is to design a Markov chain such that (i) the Markov chain is irreducible (i.e., any two states are reachable from each other) and (ii) the detailed balance equation is satisfied (i.e., $p_{\psi}^{*}q_{\psi,\psi'} = p_{\psi'}^{*}q_{\psi',\psi}, \forall\psi, \psi' \in \mathcal{T}$). Consequently, it is allowed to set the transition rates between any two states to be zero, i.e., remove their link in underlying Markov chain, if they are still reachable from any other states. We refer to [13] for further explanation.

In practice, however, direct transition between two states means migration between two tree structures. To derive a distributed algorithm, we only allow direct transitions between two states if the current and the target trees can be transformed to each other by only one parent changing operation in one of the trees. Namely, two states ψ and ψ' are directly reachable from each other if we can construct tree ψ' by deleting an edge $(i, j) \in \xi$ from ψ and adding edge $(i, k) \in \xi$ to ψ. Now, the next step is to set the transition rate as follows:

$$q_{\psi,\psi'} = \frac{1}{\exp(\alpha)}\frac{\exp(\beta\Phi_{\psi'}^{D})}{\exp(\beta\Phi_{\psi}^{D}) + \exp(\beta\Phi_{\psi'}^{D})}, \tag{7}$$

where $\alpha \geq 0$ is a constant and $q_{\psi',\psi}$ is defined symmetrically.

C. Algorithm Design

Our goal is to realize a distributed implementation of the Markov chain proposed in the previous section. In this part, we detail our implementation.

To compute transition rate between the states, the approximate values of maximum QoAs of both current (Φ_{ψ}^{D}) and the target ($\Phi_{\psi'}^{D}$) states (trees) are required. A scheduling algorithm is designed in [5] to obtain maximum QoA. However, the input for algorithm of [5] is a tree and they consider one-hop interference model. Thus, the algorithm cannot be directly used in our setting with general network topology. We extend the algorithm in [5] namely "Waiting-Assignment" algorithm (listed as Algorithm 2 and explained in Section V-C.2).

1) The Details of Parent-Changing Algorithm: Given initial aggregation tree ψ and deadline D, we first run "Waiting-Assignment" algorithm to obtain an estimation of Φ_{ψ}^{D}. Then, based on the underlying Markov chain design and in an iterative manner, we proceed to migrate to a target aggregation tree ψ' with (probably) better $\Phi_{\psi'}^{D}$ than Φ_{ψ}^{D}. To realize this end, each sensor node individually runs "Parent-Changing" algorithm which is summarized as Algorithm 1.

Algorithm 1: "Parent-Changing" Algorithm for Node $i \in \mathcal{V}$

```
Input: $\alpha, \beta$
Output: New parent of node $i$
1  $P_i \leftarrow$ parent of node $i$
2  $\mathcal{N}_{\geq i} \leftarrow \{j : (i, j) \in \xi, W_j \geq W_i\}$
3  Node $i$ generates a timer $\tau_i \sim \exp(\lambda_i)$ with mean
   $\lambda_i = \frac{1}{|\mathcal{N}_{\geq i}|}$ and starts to count down
4  When $\tau_i$ expires, node $i$ randomly selects one of its
   neighbors $P_i' \in \mathcal{N}_{\geq i}$.
5  $\Phi_{\text{prev}} \leftarrow$ node $i$'s estimation of $\Phi_{\psi}^{D}$ in Equation (7), i.e.,
   the maximum QoA of the current tree
6  Node $i$ changes its parent to $P_i'$
7  $\Phi_{\text{next}} \leftarrow$ node $i$'s estimation of $\Phi_{\psi'}^{D}$ in Equation (7), i.e.,
   the maximum QoA of the new tree
8  With probability $q_{\psi,\psi'}$, node $i$ keeps the new tree
   configuration and with probability $1 - q_{\psi,\psi'}$ switches
   back and connects to the previous parent $P_i$
9  if $i$ changed its parent in Step 8 then
10     $P_i'$ invokes Waiting-Assignment($P_i'$, $W_{P_i'}$) algorithm
       on its sub-tree
11     $P_i$ invokes Waiting-Assignment($P_i$, $W_{P_i}$) algorithm on
       its sub-tree
12 Node $i$ refreshes the timer and begins counting down
```

The detailed description of Algorithm 1 is as follows. In Line 3, an exponentially distributed random number with mean $\lambda_i = \frac{1}{|\mathcal{N}_{\geq i}|}$ is generated as the timer value. This setting is required to ensure the convergence of the corresponding Markov chain. In Line 4, node i selects a new parent P_i' such that $W_{P_i'} \geq W_i$. This ensures that after the parent changing, the data structure still remains a tree since the new structure is not a tree only if node i chooses its new parent from its successors where all have a less waiting time than node i's waiting time. Meanwhile, this strategy is also rational because finding a new parent with a shorter waiting time decreases node i's new waiting time which probably reduces QoA. In Lines 5-7, node i temporarily changes its parent and estimates the impact of this change on the maximum QoA of data aggregation. Based on the estimation and transition rate in Equation (7), in Line 8, node i decides whether to keep its new parent or not. If the new state is established, then nodes P_i and P_i' should run "Waiting-Assignment" algorithm to update waiting time of their successors because of their sub-tree changes. "Waiting-Assignment" algorithm is designed based on proposed algorithm in [5].

2) The Details of Waiting-Assignment Algorithm: In Algorithm 2, we first obtain initial waiting times for the nodes using the algorithm of [5] which may be infeasible due to the fact that it applies one-hop interference model and assumes that there is no other link in the network than the input tree links. Then, to obtain a feasible scheduling, all parallel transmissions are considered to identify interference and the interference resolved by canceling the least valuable transmissions (Lines 4-8 in the algorithm). Put it another way, when parent changing operation causes an interference in two links, we simply prevent transmission in one of them which transmits less data compared to the other link. The value of each transmission is equal to the number of sensors whose data that is aggregated in the source node of the transmission packet.

Algorithm 2: Waiting-Assignment(P, w)

```
Input: tree $\psi$ rooted at $P$, deadline $w$
Output: A feasible scheduling in tree $\psi$
1  Run algorithm of [5] on tree $\psi$ to have initial assigned
   waiting times $\vec{W}^{\psi}$
2  Assume $X_i, i \in \{j : (j, k) \in \xi^{\psi}\}$ denotes QoA obtained
   from sub-tree rooted at node $i$ based on assigned waiting
   times
3  foreach $(i, j, k, l) \in \Big\{(i, j, k, l) : (i, j), (k, l) \in \xi^{\psi}, j = P_i, l = P_k, l \neq j, W_i^{\psi} = W_k^{\psi}\Big\}$ do
4      if $d(i, l) \leq (1 + \delta)R_C$ then
5          if $X_i \geq X_j$ then
6              Set $W_j = -1$ and $n_j = 0$
7          else
8              Set $W_i = -1$ and $n_i = 0$
```

Finally, it is worthy to note that the parameter β not only affects the accuracy of the approximation, but also with large values of β, the algorithm migrates towards better configurations more greedily, whereas it may lead to premature convergence and trap into local optimum trees.

Proposition 1: "Parent-Changing" algorithm in fact implements a time reversible Markov chain with stationary distribution in Equation (4).

Proof: The designed Markov chain is finite state space ergodic Markov chain where each tree configuration in state space is reachable from any other state by one or more parent changing process. We proceed to prove that the stationary state of designed Markov chain is Equation (4). Let $\psi \rightarrow \psi'$ denote transition from state ψ to ψ' at a timer expiration and

$A = \frac{1}{\exp(\alpha)} \frac{\exp(\beta\Phi^D_{\psi'})}{\exp(\beta\Phi^D_{\psi})+\exp(\beta\Phi^D_{\psi'})}$. Moreover, $\Pr(\psi \to \psi')$ is the probability of this transition.

This probability can be calculated as follows:

$$\begin{aligned} &\Pr(\psi \to \psi') \\ &= \Pr(i \text{ chooses } P'|i\text{'s timer expires}).\Pr(i\text{'s timer expires}) \\ &= \frac{A}{|\mathcal{N}_{\geq i}|}.\frac{|\mathcal{N}_{\geq i}|}{\sum_{j\in\mathcal{V}}|\mathcal{N}_{\geq j}|} = \frac{A}{\sum_{j\in\mathcal{V}}|\mathcal{N}_{\geq j}|} \end{aligned} \tag{8}$$

In the algorithm, node i counts down with rate $|\mathcal{N}_{\geq i}|$. Therefore, the rate of leaving state ψ is $\sum_{j\in\mathcal{V}}|\mathcal{N}_{\geq j}|$. We can calculate transition rate $q_{\psi,\psi'}$ as follows:

$$q_{\psi,\psi'} = \sum_{j\in\mathcal{V}} |\mathcal{N}_{\geq j}|.\frac{A}{\sum_{j\in\mathcal{V}}|\mathcal{N}_{\geq j}|} = A. \tag{9}$$

We can see that $p^*_{\psi}.q_{\psi,\psi'} = p^*_{\psi'}.q_{\psi',\psi}$. Therefore, the detailed balance equation holds and the stationary distribution of constructed Markov chain is Equation (4) [27]. □

"Parent-Changing" algorithm is distributed if we can estimate Φ_{next} and Φ_{prev} in the algorithm in a distributed manner. By exact calculation of these values, the designed Markov chain will converge to stationary distribution in Equation (4). Hence, "Parent-Changing" algorithm can give us a near-optimal solution of the TCSP. However, exact calculation of Φ_{next} and Φ_{prev} is not possible in nodes locally since they can only be calculated in the sink. Therefore, we need to estimate their values. We estimate the values by two different methods.

Approx-1 First Method of Estimating Φ_{next} and Φ_{prev}: When node i wants to modify its parent from P_i to P_i' (and subsequently tree ψ to ψ'), one possible way of estimation is running "Waiting-Assignment" algorithm by nodes P_i and P_i' on their sub-trees. Let $\Phi_{\text{prev}}[s]$ and $\Phi_{\text{next}}[s]$ denote the maximum achievable QoAs in a sub-tree rooted at node s before and after the sub-tree change, respectively. Then, we have the following estimation:

$$\Phi_{\text{next}} \approx (\Phi_{\text{next}}[P_i] + \Phi_{\text{next}}[P_i']), \tag{10}$$

$$\Phi_{\text{prev}} \approx (\Phi_{\text{prev}}[P_i] + \Phi_{\text{prev}}[P_i']). \tag{11}$$

When node i changes its parent from P_i to P_i', only sub-trees rooted at P_i and P_i' change and all other parts of the tree remain intact and so the estimation accuracy is expected to be high. This estimation comes with the overhead of running "Waiting-Assignment" algorithm at nodes P_i and P_i' to approximate Φ_{next} and Φ_{prev}.

Approx-2: Second Method of Estimating Φ_{next} and Φ_{prev}: Another way of estimation is just using waiting times of nodes P_i' and i:

$$\Phi_{\text{next}} \approx W_{P_i'}, \tag{12}$$

$$\Phi_{\text{prev}} \approx W_i. \tag{13}$$

A larger value of $W_{P_i'}$ indicates that node i *probably* will be assigned a greater waiting time if it joins to sub-tree of P_i' and vice versa. In Section VII, we evaluate the efficiency of both mentioned methods by simulation.

D. Perturbation Analysis

In "Parent-Changing" algorithm, if we obtain the accurate value of Φ^D_{ψ} to calculate transition rates, the designed Markov chain converges to the stationary distribution given by Equation (4). Thus, we have a near-optimal solution of the TCSP with optimality gap determined in Equation (6). In distributed fashion, however, we estimate the optimal tree-specific QoAs by Equations (10), (11), (12), and (13). Consequently, the designed Markov chain may not converge to the stationary distribution in Equation (4). Fortunately, our employed theoretical approach can provide a bound on the optimality gap due to the perturbation errors of the inaccurate estimation using a quantization error model.

We assume that in a tree configuration ψ, the corresponding perturbation error is bounded to $[-\Delta_{\psi}, \Delta_{\psi}]$. In order to simplify the approach, we further assume that Φ^D_{ψ} takes only one of the following $2n_{\psi}+1$ values:

$$\begin{aligned} [\Phi^{\psi}_D - \Delta_{\psi}, \ldots, \Phi^{\psi}_D - \frac{1}{n_{\psi}}\Delta_{\psi}, \Phi^{\psi}_D, \Phi^{\psi}_D \\ + \frac{1}{n_{\psi}}\Delta_{\psi}, \ldots, \Phi^{\psi}_D + \Delta_{\psi}], \end{aligned} \tag{14}$$

where n_{ψ} is a positive constant. Moreover, with probability $\eta_{j,\psi}$, the maximum quality of aggregation is equal to $\Phi^D_{\psi} + \frac{j}{n_{\psi}}\Delta_{\psi}, \forall j \in \{-n_{\psi}, \ldots, n_{\psi}\}$ and $\sum_{j=-n_{\psi}}^{n_{\psi}} \eta_{j,\psi} = 1$.

Let $\tilde{p}$ denote the stationary distribution of the *states* in the *perturbed* Markov chain [27]. We also denote stationary distribution of the *configurations* in the original and perturbed Markov chains by $p^* : \{p^*_{\psi}, \psi \in \mathcal{T}\}$ and $\bar{p} : \{\bar{p}_{\psi}, \psi \in \mathcal{T}\}$, respectively. Then, we have [27]

$$\tilde{p} = ngleq[\tilde{p}_{\psi,\Phi^D_{\psi}+\frac{j}{n_{\psi}}\Delta_{\psi}}, j \in \{-n_{\psi}, \ldots, n_{\psi}\}, \psi \in \mathcal{T}], \tag{15}$$

$$\bar{p}_{\psi}(\Phi) = \sum_{j\in\{-n_{\psi},\ldots,n_{\psi}\}} \tilde{p}_{\psi,\Phi^D_{\psi}+\frac{j}{n_{\psi}}\Delta_{\psi}}, \forall\psi \in \mathcal{T}. \tag{16}$$

Using total variance distance [30] we can measure the distance of p^*_{ψ} and $\bar{p}_{\psi}$ as

$$d_{TV}(p^*, \bar{p}) = \frac{1}{2}\sum_{\psi\in\mathcal{T}} |p^*_{\psi} - \bar{p}_{\psi}|. \tag{17}$$

Theorem 3: a) *The total variance distance between p^*_{ψ} and $\bar{p}_{\psi}$ is bounded by* $[0, 1 - \exp(-2\beta\Delta_{max})]$ *where* $\Delta_{max} = \max_{\psi\in\mathcal{T}} \Delta_{\psi}$. b) *By defining* $\Phi_{\max} = \max_{\psi\in\mathcal{T}} \Phi^D_{\psi}$, *the optimality gap in* $|p^* - \bar{p}|$ *is*

$$|p^* - \bar{p}| \leq 2\Phi_{\max}(1 - \exp(-2\beta\Delta_{max})). \tag{18}$$

For proof and remarks, we refer to [27].

Finally, we highlight that the convergence (mixing time) of the algorithm based on Markov approximation framework is studied in [27] and [31]. Overall, this framework, in its basic setting suffers from slow rate of convergence. To mitigate this, a promising solution is to find a "good" initial aggregation tree as the input for the iterative Algorithm 1, thereby its convergence time can be improved. In this regard, the goal in the next section is to develop an algorithm to construct a fast initial aggregation tree.

VI. Initial Tree Construction Algorithm

We proceed to develop an initial tree construction algorithm to identify a good starting feasible solution for bootstrapping the Markov approximation-based algorithms. The intuition is that if Algorithm 1 starts from a tree which already has a good quality, not only high-quality data aggregation experience can be provided starting from the beginning, but also fast convergence of the algorithm proposed in the previous section can be achieved. Algorithm 3 outputs a near optimal feasible aggregation tree which is a spanning tree over the underlying WSN topology.

Algorithm 3: "FastInitTree"

```
Input: Graph $\mathcal{G} = \{\mathcal{V} \cup \{S\}, \mathcal{E}\}$, Deadline $D$
Output: Close-to-optimal spanning tree
1 Define $\mathcal{V}_{\mathsf{done}}$ as the set of nodes that their parent in final
  tree is identified
2 $\mathcal{V}_{\mathsf{done}} \leftarrow \emptyset$
3 ExtendTree($\mathcal{G}$, $S$, $D$)
4 For any node not in $\mathcal{V}_{\mathsf{done}}$ assign it to one of its
  neighbors in $\mathcal{V}_{\mathsf{done}}$ with the least number of children
```

Algorithm 4: "Extend-Tree"

```
Input: Graph $\mathcal{G} = \{\mathcal{V} \cup \{S\}, \mathcal{E}\}$, Parent $P$, Deadline $D$
Output: A tree rooted at parent $P$
1 Define $\mathcal{V}_{\mathsf{done}}$ as set of nodes that assigned to a parent
  (initially, $\mathcal{V}_{\mathsf{done}} = \emptyset$)
2 $\mathcal{V}_{\mathsf{done}} \leftarrow \mathcal{V}_{\mathsf{done}} \cup P$
3 $\mathcal{V}_{\mathsf{curr}} \leftarrow$ neighbors of $P$ except those in $\mathcal{V}_{\mathsf{done}}$
4 $k \leftarrow |\mathcal{V}_{\mathsf{curr}}|$
5 $power_i \leftarrow$ number of neighbors of $i$ in $\mathcal{V} \backslash \mathcal{V}_{\mathsf{done}}$
6 w.l.g assume that $c_1, \ldots, c_k$ are the members of $\mathcal{V}_{\mathsf{curr}}$
  sorted in a descending order based on
  $power_i, i = 1, \ldots, k$
7 for i=1:min{k,D} do
8     Set $P$ as parent of $c_i$ in the output tree
9     if $D > 0$ then
10        ExtendTree($\mathcal{G}, c_i, D - i$)
```

In a nutshell, Algorithm 3 aims to find an appropriate unique parent for each node (except the sink) in the graph. By doing so, a feasible aggregation tree is indeed constructed. In particular, $\mathcal{V}_{\mathsf{done}}$ includes the nodes that their parents are chosen and initially defined as an empty set. Algorithm 3 calls Algorithm 4 on the sink node and receives a tree rooted at the sink. However, the returned tree by Algorithm 4 may not be a spanning tree, i.e., some nodes may not be still in $\mathcal{V}_{\mathsf{done}}$. To construct the spanning tree, Algorithm 3 goes through the nodes that are not in $\mathcal{V}_{\mathsf{done}}$ and set them as children of one of their neighbors in $\mathcal{V}_{\mathsf{done}}$ with the least number of children (in Line 4 of *FastInitTree*). The intuition behind this is that a parent with less children is more likely to be able to participate its new child in the aggregation. Before proceeding to explain the details of Algorithm 4, we give definition of "*well-structured*" graph in the context of deadline-constrained aggregation tree construction, which is useful in the discussion of the algorithm.

Definition 1: Graph $\mathcal{G} = \{\mathcal{V} \cup \{S\}, \mathcal{E}\}$ with $V = |\mathcal{V}|$ is well-structured under a specific sink deadline D if the optimal tree in $\mathcal{G}$ is an optimal tree in a complete graph with V nodes where $V \geq 2^D - 1$.

Algorithm 4 is the main part of the *FastInitTree* algorithm. It works recursively on the input parent P to develop a tree. Note that the maximum number of participant nodes with deadline D is $2^D - 1$ (see [11] for the proof) that is achievable if the graph is *well-structured* for deadline D, thereby its structure allows to construct a tree with the maximum feasible QoA of $2^D - 1$. We emphasize that if a graph is well-structured, it does not imply that the number of communication links in the graph is equal (or even close) to the number of links in the corresponding complete graph. For further illustration, consider the well-structured graph in Fig. 2a with 7 (i.e., $2^3 - 1$) nodes and 10 links. Despite more connectivity among the nodes in the graph of Fig. 2b, it is not a well-structured graph and its optimal QoA is less than the optimal QoA in Fig. 2a. Finally, we call the optimal tree of a well-structured graph $\mathcal{G} = \{\mathcal{V} \cup \{S\}, \mathcal{E}\}$ as *ideal* tree for deadline D while its maximum QoA is $2^D - 1$.

Now, we turn back to explain the main idea of Algorithm 3. In Algorithm 3 we assume that the network graph is well-structured and try to build an aggregation tree such that its structure is as close as possible to the corresponding *ideal* tree. It is not difficult to see that in *ideal* tree, the number of children of each node (including sink) is equal to its waiting time (as in Fig. 2a). Based on this fact, Algorithm 3 starts from the sink node and by calling Algorithm 4 tries to find top D most *powerful* neighbors of sink, where the *power* of a node is defined as the number of its neighbors (Line 5 in Algorithm 4). Indeed, the algorithm assumes that these D nodes will have waiting times $\{D-1, \ldots, 0\}$ according to their ability to communicate with the other nodes. Then, the algorithm considers these nodes as the sink's children in the final tree. Algorithm 4 is called recursively on sink's children to build the rest of the tree. The wisdom of the algorithm in selecting children of each node makes it as a promising method.

A. Discussions on the Optimality and Complexity of Algorithm 3

Theorem 4 proves that Algorithm 3 outputs the optimal tree, given that the underlying graph is complete.

Theorem 4: Algorithm 3 generates an optimal aggregation tree given that the underlying graph is a complete graph $\mathcal{G}_k$ with $k \geq 2^D - 1$.

Proof: Note that in an *ideal* tree for deadline D, we have this key property that after running optimal scheduling algorithm [5] on the tree, each node having waiting time $w, 0 \leq w \leq D$, has exactly w children with waiting times $\{w-1, \ldots, 0\}$. By following the steps of Algorithm 3, it can be seen that the algorithm preserves the key property of the ideal tree while constructing the aggregation tree. The recursive tree construction process starts from the sink node. Since we have

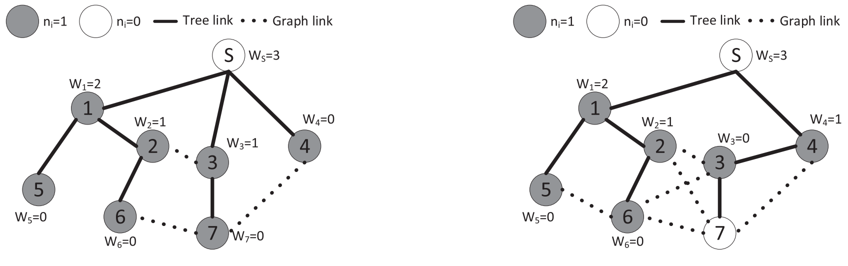


Fig. 2. Illustration of *well-structured* graph with sink deadline $D = 3$. Although the graph in Fig. 2b is more rich in terms of number of links, the structure of graph does not allow to participate all nodes under any feasible tree. (a) A $well-structured$ graph with 10 links. (b) A graph that is not well-structured with 13 links.

a complete graph, Algorithm 4 is able to choose exactly D out of N nodes with highest *power* as children of the sink. In the next step, Algorithm 4 is called on these D children of the sink namely $c_1, c_2, \ldots, c_D$ with $power_i \geq power_{i+1}, i = 1, \ldots, D-1$ and chooses $D-i$ children for $c_i, i = \{1, \ldots, D\}$ to follow the property of the ideal tree for deadline D. This process, while keeping the key property of the corresponding ideal tree, continues recursively in the same manner on the remaining nodes until all nodes assigned to a parent. Thus, it ensures that the final output is an ideal tree. □

Theorem 5: The time complexity of Algorithm 3 is $O(Nv \log v)$ where v is the maximum node degree.

Proof: The cost of Algorithm 4 is determined by total sorting cost of children for $O(N)$ nodes which is bounded by $O(Nv \log v)$. Algorithm 3 goes over nodes who are not in $\mathcal{V}_{\mathsf{done}}$ to assign them to a parent which costs $O(N)$. Therefore, the total cost of the Algorithm 3 is $O(Nv \log v)$. □

B. Remarks on the Time-Varying Deadline in the Sink

Data aggregation is a periodic action in WSNs and in each period, the sink may change aggregation parameters such as deadline. When deadline changes, the previously constructed aggregation tree in the last period may not produce the same level of QoA. In this situation, a new tree structure is needed to maximize QoA under the new deadline. A naïve approach in this situation is running the tree construction algorithm with the new deadline. However, the following theorem shows that the optimal tree does not need to be changed for the case that the deadline is decreased as compared to its previous value.

Theorem 6: If all nodes are source and $\psi^\star$ is an optimal tree under the sink deadline D, then $\psi^\star$ is optimal for deadline $D', D' = 1, 2, \ldots, D-1$. In addition, the optimal scheduling can be reconstructed by reducing the previous waiting times by $D - D'$.

Proof: We prove the theorem when $\psi^\star$ is an ideal tree. For the case that $\psi^\star$ is not ideal, the proof is similar. With the ideal tree $\psi^\star$, there are $2^D - 1$ participant nodes in the tree. Each node (including sink) with waiting time w, $w \in \{0, \ldots, D\}$, has exactly w children with assigned waiting time $\{w-1, \ldots, 0\}$. We claim that if we keep the same tree for new sink deadline D' with $D' < D$ and reduce the previously assigned waiting times by $D - D'$ then, the new scheduling is feasible and the number of participant nodes is $2^{D'} - 1$, i.e., the optimal QoA, which in turn proves the optimality of the tree for deadline D'. First, the scheduling is feasible since all waiting times are reduced by a constant and so the transmissions occur in the same order as in the previous feasible scheduling.

Second, the number of participant nodes in the new scheduling, namely X, can be calculated by subtracting total number of nodes with waiting time less than D' from $2^D - 1$ since with the new scheduling, these nodes' waiting times will be negative which has no meaning and makes them non-participant nodes. Therefore, we can sum up all nodes in the previous scheduling having waiting time greater than or equal to D' to find X. Formally, we have $X = f(D-D') + f(D-D'+1) + \cdots + f(D)$ where, $f(i)$ denotes the number of nodes in the previous scheduling with assigned waiting time i. Note that we have $f(i) = f(i+1) + f(i+2) + \cdots + f(D)$ and $f(D) = f(D-1) = 1$. Then, we can calculate $\sum_{i=D-D'}^{D} f(i)$ as follow:

$$\begin{aligned}
&\overbrace{f(D-D')}^{\text{A}} + \overbrace{f(D-D'+1)}^{\text{B}} + \cdots + \overbrace{f(D)}^{\text{C}} \\
&= \overbrace{f(D-D'+1) + f(D-D'+2) + \cdots + f(D)}^{\text{A}} \\
&\quad + \overbrace{f(D-D'+2) + \cdots + f(D)}^{\text{B}} + \ldots + \overbrace{f(D)}^{\text{C}}.
\end{aligned}$$

By solving the above equation we have

$$X = \sum_{i=D-D'}^{D} f(i) = \sum_{i=0}^{D'-1} 2^i = 2^{D'} - 1.$$

□

Theorem 6 implies that if we construct a near optimal tree for a specific deadline, then the same tree can be used for all shorter deadlines. Most importantly, the new scheduling is straightforward and needs no cost. This can help to avoid the overhead of running the tree construction and scheduling algorithm when deadline changes.

VII. Simulation Results

In this section, we evaluate the proposed algorithms through simulations. Unless otherwise specified, the settings are as follows: 100 sensor nodes uniformly dispersed in a square field with side length of 300m. Sink node is located at the

TABLE II
ACRONYMS FOR THE ALGORITHMS

Notation	Description
Approx-1	Approximation algorithm that estimates the current QoA using Equations (10) and (11) in each node (has some overheads)
Approx-2	Approximation algorithm that estimates the current QoA using Equations (12) and (13) in each node (has no overheads)
Algorithm 2	Waiting-Assignment algorithm as a modified version of algorithm of [5] to consider protocol interference model
Approx-1H	Approx-1 starting from initial tree constructed by FastInitTree
Approx-2H	Approx-2 starting from initial tree constructed by FastInitTree
FastInitTree	Heuristic algorithm

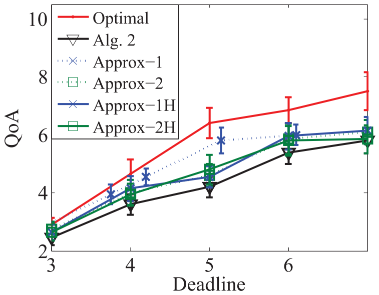


Fig. 3. Quality of aggregation vs. deadline ($V = 15$).

center of the top side of the square field i.e., its position is $(150, 300)$. The protocol interference parameter δ is 1 and communication range of each node is 75m, i.e., two nodes are connected in the network if their distance is "$\leq$ 75m". After deployment, sensor nodes construct an initial data aggregation tree. Except for the experiments that use initial tree built by *FastInitTree* algorithm, the tree is constructed based on Greedy Incremental Tree (GIT) algorithm [32]. We let $\alpha = 0.2$ and $\beta = 2$, and choose 80% of nodes randomly as the sources. Each data point of the figures belongs to the average value of 50 runs with 95% confidence interval where each run is a different random topology. Moreover, for each topology, sink imposes a deadline in terms of time slots uniformly and randomly selected from interval [10,20]. We report the results of approximation algorithms (summarized in Table II) after 50 iterations where an iteration is defined as a timer expiration of a sensor node.

A. Performance Comparison With the Optimal Solution

In this section, we compare the performance of the proposed methods to the optimal solution. Since calculating optimal solution is computationally infeasible in large scale networks, we set up a small comparison experiment where 15 sensor nodes with communication range of 10m dispersed in a field with side length of 40m and sink coordinate is (20,40). Moreover, due to small network size, we consider all sensors as the source nodes.

Fig. 3 portrays QoA of markov based algorithms against sink deadline. The main purpose is to compare our schemes with the optimal. The result for the algorithms "Approx-1", "Approx-2", "Approx-1H" and "Approx-2H" are very close to each other. "Approx-1" is 88% close to optimal in this case which is slightly better than the other algorithms. We believe that in real-world scenarios with the higher number of sensor nodes, the performance difference between the markov based algorithms is more visible than that of the small scenario. To scrutinize this claim in more detail, we set up another set of experiments to investigate the improvements against various deadlines in the next subsection.

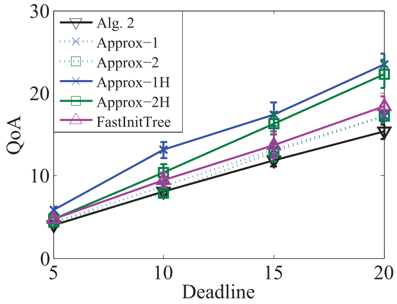


Fig. 4. Quality of aggregation vs. deadline ($V = 100$).

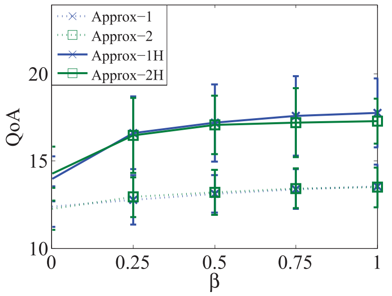


Fig. 5. Improvement in quality of aggregation vs. β.

B. The Effect of the Deadline

We study the effect of sink deadline on QoA. Based on Fig. 4, the trend is that QoA improves as deadline increases. The reason is that by increasing the deadline, more sensor nodes have the opportunity to participate in data aggregation.

A notable observation is that *FastInitTree* shows a better performance as compared to both "Approx-1" and "Approx-2". Its result is also 78% of the "Approx-1H". There is also a small difference between "Approx-1" and "Approx-2". On average, "Approx-2" is 97% close to "Approx-1". Based on these observations, *FastInitTree* seems a proper choice with respect to its low overhead and low cost. "Approx-1H" has the best performance among all algorithms. "Algorithm 2" which is based on proposed algorithm in [5] and does not change the structure of the initial aggregation tree achieved the least QoA. The poor performance of "Algorithm 2" is the direct consequence of ignoring the impact of aggregation tree.

C. The Effect of Parameter β

As it is stated in Section IV.A, the approximation gap theoretically decreases while β increases. This parameter

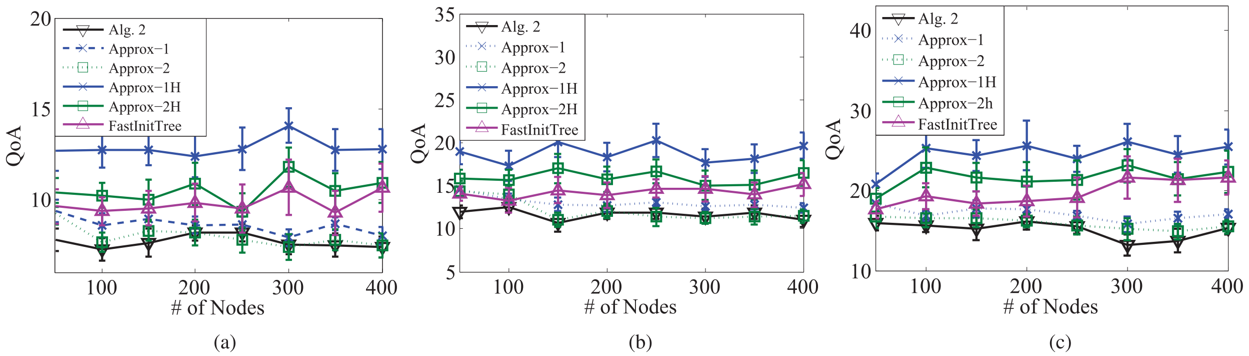


Fig. 6. Quality of aggregation vs. network size. The deployment field is same for all scenarios to study the effect of network density. (a) $D = 10$. (b) $D = 15$. (c) $D = 20$.

is an input for the proposed Markov based approximation algorithms and has a big impact on convergence rate of the algorithms. We depict the effect of β by simulation in Fig. 5. Since "Algorithm 2" and *FastInitTree* are independent of the value of β, Fig. 5 only portrays the results for Markov based algorithms. By increasing β, in addition to achieving higher QoA, we observe that the QoA momentum of Markov based schemes degrades while β grows. This is a consequence of fast convergence of approximation schemes to the optimal where in the proximity of optimal solution improvements are smaller. The experimental results of Fig. 5 confirm the theoretical analysis in Section V.

D. The Effect of the Network Size

Figs. 6a-6c depict obtained QoA values for network sizes of 50 to 400 with step 50 for deadline values of 10, 15, and 20. We fixed the deployment field while the number of nodes increases. Therefore, in these scenarios, the greater the network size, the denser the network is. The QoA in all scenarios of Fig. 6 does not change significantly with the increase in network size. Hence, we can conclude that the behavior of studied methods does not change significantly as network density increases. However, as deadline increases the obtained QoA values of different algorithms increase which is the direct consequence of the observation in Section VII-B. Another interesting observation is that in most cases *FastInitTree* outperforms "Approx-1" and "Approx-2" that start from a random tree. However, when their initial state is set to the output of the *FastInitTree* to make algorithms "Approx-1H" and "Approx-2H", the best results are achieved.

E. The Effect of FastInitTree on the Convergence Rate

Since the transition rates are set wisely to improve the maximum QoA of the aggregation tree, we expect to obtain a better QoA as the number of transitions increases. Each transition can only occur after a node's timer expiration. However, all timer expirations do not lead to a transition. A key point here is that a desired level of QoA can be achieved with a fewer number of transitions if the initial tree provided by *FastInitTree* algorithm is chosen wisely. Fig. 7 demonstrates how the four Markov-based approximation algorithms improve as the number of iterations increases. Note that the only difference between "Approx-1" and "Approx-1H" is in their initial state. This is the same for "Approx-2" and "Approx-2H". Then, a key point here is the effect of *FastInitTree* algorithm on the convergence rate of Markov approximation. "Approx-1H" with 10 iterations achieves the same QoA that "Approx-2" obtains after 40 iterations. In a similar case, "Approx-2H" with only 10 iterations works better than "Approx-2" after 40 iterations.

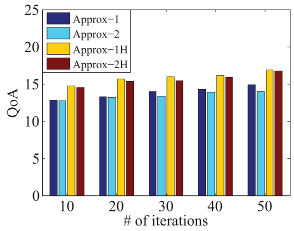


Fig. 7. QoA vs. iteration numbers.

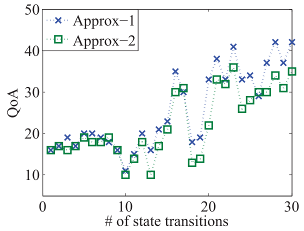


Fig. 8. Improvement of QoA for a random topology.

Finally, in a *microscopic* view in Fig. 8, we demonstrate the evolution of the maximum achieved QoA after each transition, i.e., migrating to a new aggregation tree, for a randomly selected sample topology.

VIII. Conclusion

In this paper, we addressed the NP-hard problem of constructing data aggregation tree in WSNs, with the goal of maximizing the number of nodes that the sink receives their data within an application-specific aggregation deadline. Two successive algorithms were proposed: first, a distributed algorithm that runs in iterative manner and eventually converges to a bounded neighborhood of the optimum, and second, a bootstrapping algorithm with low complexity that can be served as a good initial point for the former. Observations on experiments corroborated our analysis on the importance of constructing the optimal aggregation tree. Moreover, experimental results demonstrated that our methods achieved a close-to-optimal solution and significant performance improvement obtained by using appropriate aggregation tree. Last but not the least, this work is the first attempt on leveraging Markov approximation as a general framework to tackle tree construction in WSNs and we believe that this solution approach can be used in several other applications for constructing trees in distributed manner.

Obtained results open several important future directions. It would be interesting to incorporate energy consumption and turn the problem to an energy-aware QoA maximization one. This is important because the data aggregation is a periodic operation in the network and hence, relying on a fixed aggregation tree for a long time may lead to energy depletion of some specific nodes and degrade the network performance and lifetime. A wise policy might be to try to follow a uniform distribution of nodes' contribution in data aggregation, while keeping the QoA at the desired level. The second line is to tackle forest construction problem for multi-sink networks. This is a challenging problem, since the single sink scenario, as the special case of a multi-sink network, has been proved to be NP-hard.

Appendix

A. How to Schedule Data Aggregation Given a Fixed Underlying Tree

In this Appendix, we explain how to find the maximum QoA in a given tree using a simple and tractable example. The example also clarifies the data aggregation model.

Consider the data aggregation tree in Fig. 9 where the sink deadline is set to $D = 2$ and all nodes are source. For the ease of explanations, we assume that there is no other link in the network graph. With the given deadline, the sink can choose at most $D = 2$ children (due to interference constraint) and assign their waiting times as *distinguished* values between 0 and $D - 1 = 1$. To maximize the number of source participant nodes (QoA), one of the possible choices for the sink is the assignment of $W_1 = 0$ and $W_2 = 1$. With this assignment, node 2 can assign a waiting time of 0 to one of its children (in this example node 5 with $W_5 = 0$). Eventually, the maximum QoA is 3 and participant nodes are $1, 2, 5$. During the aggregation process, in the first time slot, node 1 and node 5 send their packets to their parents in parallel. In the second time slot, node 2 aggregates its own packet with the received data from node 5 and sends the aggregated data to the sink. It is not hard to see that this scheduling policy is optimal, i.e., it achieve the maximum QoA given the fixed aggregation tree. As a non-optimal waiting time assignment, consider the assignment of $W_1 = 1, W_2 = 0$. In this case, the final QoA is 2 with participant nodes 1 and 2. With $D = 3$, the maximum QoA is 7 and the optimal assignment is $W_1 = W_4 = W_6 = W_7 = 0$, $W_2 = 2$, $W_3 = 1$ and $W_5 = 1$.

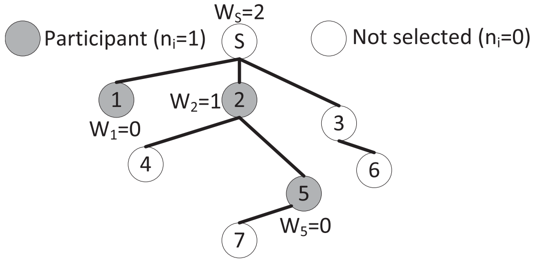


Fig. 9. An example of scheduling given a fixed underlying tree with $D = 2$. It is assumed that $d(1, 2) \geq (1 + \delta)R_C$ and $d(S, 5) \geq (1 + \delta)R_C$.

In [5], an algorithm is proposed to achieve the maximum QoA in a given tree ψ. The scheduling algorithm in [5] is optimal given a fixed tree as input and it does not change the structure of the tree for further improvement of QoA. Moreover, [5] assumes a one-hop interference model which is not suitable for graph topology.

References

[1] B. Alinia, M. H. Hajiesmaeili, and A. Khonsari, "On the construction of maximum-quality aggregation trees in deadline-constrained WSNs," in *Proc. IEEE INFOCOM*, May 2015, pp. 226–234.

[2] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Trans. Wireless Commun.*, vol. 1, no. 4, pp. 660–670, Oct. 2002.

[3] R. Rajagopalan and P. K. Varshney, "Data-aggregation techniques in sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 8, no. 4, pp. 48–63, 4th Quart., 2006.

[4] S. Javadi, M. H. Hajiesmaili, A. Khonsari, and B. Moshiri, "Temporal-aware rate allocation in mission-oriented WSNs with sum-rate demand guarantee," *Comput. Commun.*, vol. 59, pp. 52–66, Mar. 2015.

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

[6] F. Yuan, Y. Zhan, and Y. Wang, "Data density correlation degree clustering method for data aggregation in WSN," *IEEE Sensors J.*, vol. 14, no. 4, pp. 1089–1098, Apr. 2014.

[7] M. H. Hajiesmaili, M. S. Talebi, and A. Khonsari, "Multi-period network rate allocation with end-to-end delay constraints," *IEEE Trans. Control Netw. Syst.*, to be published.

[8] S. Hariharan, Z. Zheng, and N. B. Shroff, "Maximizing information in unreliable sensor networks under deadline and energy constraints," *IEEE Trans. Autom. Control*, vol. 58, no. 6, pp. 1416–1429, Jun. 2013.

[9] H. Li, C. Wu, Q. S. Hua, and F. C. M. Lau, "Latency-minimizing data aggregation in wireless sensor networks under physical interference model," *Ad Hoc Netw.*, vol. 12, pp. 52–68, Jan. 2014.

[10] X. Xu, X.-Y. Li, X. Mao, S. Tang, and S. Wang, "A delay-efficient algorithm for data aggregation in multihop wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 1, pp. 163–175, Jan. 2011.

[11] B. Alinia, H. Yousefi, M. S. Talebi, and A. Khonsari, "Maximizing quality of aggregation in delay-constrained wireless sensor networks," *IEEE Commun. Lett.*, vol. 17, no. 11, pp. 2084–2087, Nov. 2013.

[12] A. Iyer, C. Rosenberg, and A. Karnik, "What is the right model for wireless channel interference?" *IEEE Trans. Wireless Commun.*, vol. 8, no. 5, pp. 2662–2671, May 2009.

[13] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6301–6327, Oct. 2013.

[14] X. Chen, X. Hu, and J. Zhu, "Minimum data aggregation time problem in wireless sensor networks," in *Proc. 1st Int. Conf. Mobile Ad-hoc Sens. Netw. (MSN)*, 2005, pp. 133–142.

[15] P. J. Wan, S. C. H. Huang, L. Wang, Z. Wan, and X. Jia, "Minimum-latency aggregation scheduling in multihop wireless networks," in *Proc. ACM MOBIHOC*, 2009, pp. 185–194.

[16] X. Y. Li *et al.*, "Efficient data aggregation in multi-hop wireless sensor networks under physical interference model," in *Proc. IEEE MASS*, Oct. 2009, pp. 353–362.

[17] L. Guo, Y. Li, and Z. Cai, "Minimum-latency aggregation scheduling in wireless sensor network," *J. Combinat. Optim.*, vol. 31, no. 1, pp. 279–310, 2014.

[18] S. Hariharan and N. B. Shroff, "Deadline constrained scheduling for data aggregation in unreliable sensor networks," in *Proc. IEEE WiOpt*, May 2011, pp. 140–147.

[19] Z. Zheng and N. B. Shroff, "Submodular utility maximization for deadline constrained data collection in sensor networks," *IEEE Trans. Autom. Control*, vol. 59, no. 9, pp. 2400–2412, Sep. 2014.

[20] H. X. Tan, M. C. Chan, W. Xiao, P. Y. Kong, and C. K. Tham, "Information quality aware routing in event-driven sensor networks," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[21] Y. Wu, Z. Mao, S. Fahmy, and N. B. Shroff, "Constructing maximum lifetime data-gathering forests in sensor networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 5, pp. 1571–1584, 2010.

[22] D. Li, J. Cao, M. Liu, and Y. Zheng, "Construction of optimal data aggregation trees for wireless sensor networks," in *Proc. IEEE ICCCN*, Mar. 2006, pp. 1571–1584.

[23] Y. Wu, S. Fahmy, and N. B. Shroff, "On the construction of a maximum-lifetime data gathering tree in sensor networks: NP-completeness and approximation algorithm," in *Proc. IEEE INFOCOM*, Apr. 2008, pp. 356–360.

[24] S. Wan, Y. Zhang, and J. Chen, "On the construction of data aggregation tree with maximizing lifetime in large-scale wireless sensor networks," *IEEE Sensors J.*, vol. 16, no. 20, pp. 7433–7440, Oct. 2016.

[25] T. W. Kuo and M. J. Tsai, "On the construction of data aggregation tree with minimum energy cost in wireless sensor networks: NP-completeness and approximation algorithms," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 2591–2595.

[26] M. Shan, G. Chen, D. Luo, X. Zhu, and X. Wu, "Building maximum lifetime shortest path data aggregation trees in wireless sensor networks," *ACM Trans. Sensor Netw.*, vol. 11, no. 1, 2014, Art. no. 11.

[27] S. Zhang, Z. Shao, M. Chen, and L. Jiang, "Optimal distributed P2P streaming under node degree bound," *IEEE/ACM Trans. Netw.*, vol. 22, no. 3, pp. 717–730, Jun. 2014.

[28] M. H. Hajiesmaili, L. T. Mak, Z. Wang, C. Wu, M. Chen, and A. Khonsari, "Cost-effective low-delay cloud video conferencing," in *Proc. IEEE ICDCS*, Jul. 2015, pp. 103–112.

[29] C. Chekuri and A. Kumar, "Maximum coverage problem with group budget constraints and applications," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2004, pp. 72–83.

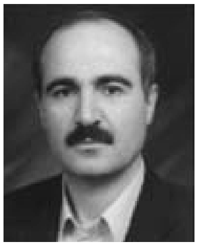
[30] P. Diaconis and D. Stroock, "Geometric bounds for eigenvalues of Markov chains," *Ann. Appl. Probab.*, vol. 1, no. 1, pp. 36–61, 1991.

[31] S. Ziyu, X. Jin, W. Jiang, M. Chen, and M. Chiang, "Intra-data-center traffic engineering with ensemble routing," in *Proc. INFOCOM*, 2013, pp. 2148–2156.

[32] B. Krishnamachari, D. Estrin, and S. Wicker, "Modelling data-centric routing in wireless sensor networks," in *Proc. IEEE INFOCOM*, Jun. 2002, pp. 39–44.

Bahram Alinia received the master's degree in information technology from the University of Tehran, Tehran, Iran. He is currently pursuing the Ph.D. degree with the Architecture Laboratory, Institute Telecom SudParis, France. He has been a contributor to the ITEA European projects in the domain of smart grids since 2014. His research area includes wireless communications, energy systems, and approximation.

Mohammad H. Hajiesmaili received the B.Sc. degree from the Department of Computer Engineering, Sharif University of Technology, Iran, in 2007, and the M.Sc. and Ph.D. degrees from the Electrical and Computer Engineering Department, University of Tehran, Iran, in 2009 and 2014, respectively. He was a Researcher with the School of Computer Science, Institute for Research in Fundamental Sciences, Iran, from 2008 to 2013, and a Post-Doctoral Fellow with the Department of Information Engineering, Chinese University of Hong Kong, from 2014 to 2016. He is currently a Post-Doctoral Fellow with the Department of Electrical and Computer Engineering, Johns Hopkins University. His research interests include optimization, algorithm, and mechanism design in energy systems, electricity market, transportation networks, and multimedia.



Ahmad Khonsari received the B.Sc. degree in electrical and computer engineering from Shahid-Beheshti University, Iran, in 1991, and the M.Sc. degree in computer engineering from IUST, Iran, in 1996, and the Ph.D. degree in computer science from the University of Glasgow, U.K., in 2003. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Tehran, Iran, and a Researcher with the School of Computer Science, IPM, Iran. His research interests are performance modeling/evaluation, wired/wireless networks, distributed systems, and high-performance computer architecture.

Noel Crespi (SM'16) received the master's degree from ENST and the Ph.D. and Habilitation degrees from Paris VI University. In 1993, he was with CLIP, Bouygues Telecom. He joined the France Telecom Research and Development in 1995, where he led the Prepaid Service Project and took an active role in various standardization committees. In 1999, he joined Nortel Networks as a Telephony Program Manager. He joined the Institut Telecom in 2002, where he is currently a Professor and the Program Director leading the Network and Service Architecture Group. He was appointed as a Coordinator for the standardization activities in ETSI and 3GPP. He is also a Visiting Professor with the Asian Institute of Technology.