Efficient Scheduling for Periodic Aggregation Queries in Multihop Sensor Networks

XiaoHua Xu, *Student Member, IEEE*, Xiang-Yang Li, *Senior Member, IEEE, Member, ACM*, Peng-Jun Wan, and ShaoJie Tang, *Student Member, IEEE*

***Abstract*—In this paper, we study periodic query scheduling for data aggregation with minimum delay under various wireless interference models. Given a set $\mathcal{Q}$ of periodic aggregation queries, each query $Q_i \in \mathcal{Q}$ has its own period p_i and the subset of source nodes $\mathcal{S}_i$ containing the data. We first propose a family of efficient and effective real-time scheduling protocols that can answer every job of each query task $Q_i \in \mathcal{Q}$ within a relative delay $O(\mathrm{p}_i)$ under resource constraints by addressing the following tightly coupled tasks: routing, transmission plan constructions, node activity scheduling, and packet scheduling. Based on our protocol design, we further propose schedulability test schemes to efficiently and effectively test whether, for a set of queries, each query job can be finished within a finite delay. Our theoretical analysis shows that our methods achieve at least a constant fraction of the maximum possible total utilization for query tasks, where the constant depends on wireless interference models. We also conduct extensive simulations to validate the proposed protocol and evaluate its practical performance. The simulations corroborate our theoretical analysis.**

***Index Terms*—Aggregation, delay, interference, periodic, query scheduling, schedulability, utilization.**

I. Introduction

WIRELESS sensor networks (WSNs), in which different types of sensors collaborating to monitor physical or environmental conditions, are being widely used in cyber-physical systems [13], [22] to provide query services. In response to query requests of a control application, the corresponding sensory data need to be streamed to the control center. In contrast to raw data collection, in-network aggregation can reduce the requirements for both network bandwidth and power consumptions while guaranteeing the validities of the aggregated data for answering queries.

Manuscript received August 20, 2010; revised March 23, 2011 and June 12, 2011; accepted August 10, 2011; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor G. Xue. Date of publication September 08, 2011; date of current version June 12, 2012. This work was supported in part by the NSF under Grants CNS-0832120 and CNS-1035894, the program for Zhejiang Provincial Key Innovative Research Team, the program for Zhejiang Provincial Overseas High-Level Talents (One-hundred Talents Program), the National Basic Research Program of China (973 Program) under Grants 2010CB328100 and 2010CB334707, and the Tsinghua National Laboratory for Information Science and Technology (TNList).

X. Xu, P.-J. Wan, and S. Tang are with the Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616 USA (e-mail: xxu23@iit.edu; wan@cs.iit.edu; stang7@iit.edu).

X.-Y. Li is with the Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616 USA, and also with the Tsinghua National Laboratory for Information Science and Technology (TNList), Beijing 100084, China, and HangZhou DianZi University, Hangzhou 310018, China (e-mail: xli@cs.iit.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2011.2166165

The majority of work, e.g., [4], [8], [19], [20], [23], and [25], for data aggregation scheduling focused on the problem of so-called single *One-shot* Query Scheduling (OQS), where each node v_i in the network contains only one data item, say x_i, and the control center is only interested in getting the value $f(x_1, x_2, \ldots, x_n)$ for some aggregation function $f()$ (e.g., $\min$, average, or variance) with minimum delay. However, in many practical systems, query requests often come in a *periodic* fashion. For example, a query in a structural health monitoring system may request the sensory data of vibration periodically. Moreover, multiple queries may be performed simultaneously in the network, and queries may differ in many aspects, e.g., some query asks for the average temperature, while another asks for the monitored video in the same local area. On the other hand, for mission-critical real-time systems, the semantics and the validities of data often highly depend on the time of utilizing the data. For example, a surveillance system may require the positions of an intruder to be reported to the control center within a delay of seconds so that pursuing actions can be initiated in time. Then, queries are often subject to stringent delay constraints, in addition to their already complex appearances (i.e., periodic and multiple queries).

We will study the following problem. Given a WSN consisting of a set of sensor nodes and the control center (or sink node), the sink will issue to the network a set $\mathcal{Q}$ of periodic aggregation queries. For each query $Q_i \in \mathcal{Q}$, the sink may be interested in data only from a certain region, and thus only a subset of nodes will generate data to satisfy the query. We call these nodes source nodes. For each period of each query, the sink expects to receive the corresponding (possibly aggregated) data from the source nodes. For a given wireless interference model (we do not restrict ourself to a specific interference model), this objective is to jointly design a routing tree for each query and an interference-free schedule of activities for all nodes (i.e., when to transmit and which packet to transmit) such that for each query Q_i, every job can be answered within a finite delay (typically a constant multiple of its period). We note the problem as *Periodic Aggregation Query Scheduling* (PAQS); its formal definition is available in Section II-B. PAQS has been proven to be NP-hard even in its simplest form (i.e., the problem OQS) [4]. In addition, solutions for OQS cannot serve directly as a basis for solving the problem PAQS after comparing these two problems as follows.

PAQS Versus OQS: While the problem PAQS shares the notion of delay bounded in-network aggregation scheduling with OQS, these two problems differ in three aspects. First, the prerequisites of two problems are different. For OQS, there are *no* activities scheduled in advance at any node before running the scheduling protocol. However, for PAQS, this prerequisite was

not satisfied from the second query: When we schedule the ith query ($i \geq 2$), nodes already have lots of time-slots reserved to serve first $(i-1)$ queries.

Second, the objectives of two problems are different. For OQS, since each node only needs to transmit once, this query can always be satisfied. The matter is to minimize the delay of answering this query. However, for periodic queries, they may not be satisfied within finite delay. Even if there is only one node (one-node network example is the widely studied real-time scheduling problem in real-time community), when the request rates of queries/tasks become large, the delay for answering some task(s) will race toward infinity. That is because in a system of periodic tasks, larger request rates imply larger total utilization; when the total utilization of all tasks exceeds the schedulable utilization of any algorithm ("capacity" of the network), the system is not schedulable [12]. As an illustration, let us consider a one-node network instance with two periodic tasks $\{Q_1, Q_2\}$. Assume each task has a period of one time-slot, and requests exactly one time-slot to process for each period. Observe that both periodic tasks have the request rate of one; the total utilization is two, which exceeds one. Then, the network instance is overloaded with processing for these two tasks $\{Q_1, Q_2\}$; this fact will result in an infinite delay for at least one task. For periodic queries in the problem PAQS, the objective is to satisfy as many queries/tasks as possible instead.

Third, the *conflict constraints* imposed on these two problems are different. For one-shot query, since each node only needs to transmit data once, we only need to worry about the interferences from nearby nodes ("intraquery spatial constraints"). However, for multiple queries, we need to account for additional constraints: 1) for any node, the time-slots scheduled for one period of some query cannot overlap with the time-slots scheduled for another period of the same query or for another query ("interperiod node constraints" and "interquery node constraints"); 2) the time-slots scheduled for any node to answer one query cannot overlap with the time-slots scheduled for some nearby nodes to answer any query ("interquery spatial constraints" and "intraquery spatial constraints").

Our Main Contributions: Due to unique challenges for PAQS, we will propose a novel design of scheduling protocols to orchestrate both the real-time job scheduling and in-network aggregation for answering given queries. For a set of periodic data aggregation queries, we design a family of routing, node-, and packet-level *scheduling protocols* under various wireless interference models such that each query can be satisfied (the sink node can receive all the data for each query), within a bounded end-to-end delay. Our main idea is to split the sensor network spatially and temporally and find a schedule that makes efficient and careful use of resources. We prove that our protocol can achieve a total load that is at least a constant fraction of the optimum load. At the same time, for each query, the delay is at most a small constant factor of the minimum delay by which any protocol can achieve.

Our second main contribution lies in *schedulability test* schemes that test whether a given set of periodic aggregation queries can be satisfied using any possible method. We propose necessary conditions for schedulability (summarized in Theorem 6), such that if a set of queries in a network does not satisfy the conditions, we can determine immediately that the network is overloaded with query tasks (or the total request rate of all queries exceeds the "capacity" of the network). We also propose sufficient conditions for schedulability (summarized in Theorem 5) based on our protocol design. The gap between the proposed sufficient conditions and necessary conditions is proved to be a constant. This implies that the proposed sufficient conditions can achieve a utilization that is at least a constant fraction of the optimum utilization for schedulability. In addition to theoretical analysis, we conduct extensive simulation studies on our protocol design and schedulability test, the result of which corroborates our theoretical analysis.

The rest of the paper is organized as follows. Section II formulates the query scheduling problem in WSNs. Section III reviews the related work and introduces our preliminary results. In Section IV, we present our protocol design for scheduling periodic aggregation queries under various interference models. In Section V, we propose schedulability test schemes for a set of periodic aggregation queries. We present our simulation results in Section VI. Section VII discusses the limitation of this work and possible future work. Section VIII concludes the paper.

II. Model, Problem Formulation

A. System Model

Let $G = (V, E)$ represent a WSN consisting of a set V of n nodes and a set E of bidirectional communication links. Let $v_s \in V$ be the sink node. There exists a communication link between two nodes iff they are within the transmission range of each other. To transmit data, we assume TDMA scheduling where the time domain is divided into time-slots of fixed length, and the transmission of each packet costs one time-slot. Note that in practice, the time granularity is a frame that consists of multiple time-slots. A node will be continuously transmitting multiple packets for a whole frame as an atomic action. For simplicity, we assume time granularity is a time-slot here.

For wireless communications, we have to avoid interferences. In the wireless network community, several interference models have been commonly adopted, e.g., *Protocol Interference Model* (PrIM), *RTS/CTS Model*, and *Physical Interference Model* (PhIM). In PrIM [7], each node has a fixed *transmission range* normalized to one and a fixed *interference range* of ρ. Any node $v \in V$ will be interfered by the signal from another node $u \in V$ if $\|uv\| \leq \rho$ and the node v is *not* the intended receiver of the transmission from u. In the RTS/CTS model [1], for every pair of active transmitter and receiver, any other node that lies within the interference range of either the transmitter or the receiver cannot transmit simultaneously. In PhIM [3], [21], there is a threshold value $\beta > 0$, such that a node $v \in V$ can correctly receive the data from a sender u iff *the signal-to-interference-plus-noise ratio* (SINR) $\mathrm{SINR} = \frac{P_u \cdot \|uv\|^{-\kappa}}{\xi + \sum_{w \in I} P_w \cdot \|wv\|^{-\kappa}} \geq \beta$, where $\|uv\|$ is the Euclidean distance between the nodes u and v, $\xi > 0$ is the background Gaussian noise, while I is the set of other actively transmitting nodes when node u is transmitting, $\kappa > 2$ is the path loss exponent, and $P_u = P$, $\forall u \in V$ is the uniform transmission power of each node u. We observe that if a communication link has length close to $\mathbb{L} = \sqrt[\kappa]{\frac{P}{\xi\eta}}$, then the transmission of this link is prone to fail. Thus, under PhIM, we will focus on a subgraph of G, with edge length at most $\delta \cdot \mathbb{L}$,

where $\delta < 1$ is a constant (see [24] for details), and construct routing trees in this subgraph instead.

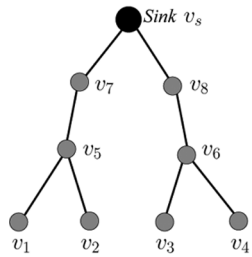
Assume the control application issues a set of query tasks $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_m\}$. For query $Q_i \in \mathcal{Q}$, let $\mathbf{p}_i$ be its period, $\mathcal{S}_i \subseteq V$ be the set of source nodes that contain data for answering query Q_i, and each source node $v \in \mathcal{S}_i$ will generate a data unit in every period to be gathered to the sink v_s. We assume that it will take χ_i time to transmit one data unit for query Q_i over any communication link. For different queries, the size of data units may be different. Then, χ_i may be different. For Q_i, let $\mathbf{a}_i$ represent the release time (or phase), and let $\mathbf{d}_i$ represent the end-to-end *delay requirement* for receiving the answer. Then, the jth instance, denoted as $\mathbf{q}_{i,j}$, of this query will be released at time $\mathbf{a}_i + (j-1) \cdot \mathbf{p}_i$, and the deadline for the sink to receive the answer is $\mathbf{a}_i + (j-1) \cdot \mathbf{p}_i + \mathbf{d}_i$.

We will focus on data aggregation queries. Data aggregation allows in-network fusion of data from different sensors when en-routing data toward the sink. We implicitly assume that the clocks of different nodes are synchronized such that only data from the same period of the same query are allowed to be aggregated. For simplicity, we assume that a node can aggregate multiple incoming data units into a single outgoing data unit of the *same* size.

For some query $Q_i \in \mathcal{Q}$, the data unit generated by each node may be large. Thus we need to split the data unit into multiple packets, then aggregate one packet at a time. For example, consider a query where each data unit can be split into two packets: One is for the temperature; the other is for the humidity. Then, a node can perform aggregation on the packets for temperature first, when it received the corresponding packets, and then perform aggregation on the packets for humidity later for this query. A packet for temperature cannot simply be aggregated with a packet for humidity. We assume that $\chi_i/\mathbf{t}_0$ is an integer, where χ_i is the processing time for query Q_i, and $\mathbf{t}_0$ is the time duration of a time-slot. When transmitting multiple packets generated by a node in one period, we allow the perfect *preemption*, which means that we can interleave transmissions of packets originated from different queries or from different periods of the same query. However, we assume that preemption only happens after a time-slot is finished since we assume a time-slot is an atomic, indivisible time unit. For simplicity, we assume that $\mathbf{p}_i$ is an integer and the actual value of a query Q_i's period is $\mathbf{p}_i\mathbf{t}_0$.

B. Problem Formulation

Given a set of preemptive, independent, and periodic aggregation query tasks, Here, "independent" means that requests for a certain task do not depend on the initiation or the completion of requests for other tasks [11]. The first objective is to design a routing structure and a transmission schedule to answer all queries and meet the delay requirement of each query. Here, the *routing structure* consists of a set of routing trees $\{T_i : 1 \le i \le m\}$, one tree T_i for each query Q_i. A *transmission schedule*, denoted as $\mathcal{S}$, consists of assigned time-slots to transmit for packets at each node. Let $t_u^{\langle \mathbf{q}_{i,j}, p \rangle}$ be the time-slot assigned to the pth packet of a job $\mathbf{q}_{i,j}$ of query Q_i at a node u. A transmission schedule is said to answer a set of queries (or check if there is a *feasible* schedule) iff the sink can receive all data for every period of each query, when every packet is transmitted according to the schedule and routed via the routing tree.



Schedule

Node(s)	Time-Slot
v_1, v_3	$3k+1$
v_2, v_4	$3k+2$
v_5, v_6	$3k+3$
v_7	$3k+4$
v_8	$3k+5$

Fig. 1. Example for periodic query scheduling.

Given a feasible schedule $\mathcal{S}$, the *end-to-end delay of a job* $\mathbf{q}_{i,j}$ in a query task Q_i (denoted as $D(\mathcal{S}, Q_i, j)$) is defined as the lapse of time-slots from the time-slot when this job $\mathbf{q}_{i,j}$ is released to the time-slot when the sink node received the final aggregated value for this job, i.e., $D(\mathcal{S}, Q_i, j) = \max_u \max_p \{t_u^{\langle \mathbf{q}_{i,j}, p \rangle} - (\mathbf{a}_i + (j-1) \cdot \mathbf{p}_i)\}$. The *end-to-end delay* of query task Q_i under the schedule $\mathcal{S}$ is defined as $\max_j D(\mathcal{S}, Q_i, j)$, which is required to be at most the delay requirement $\mathbf{d}_i$ for Q_i. We assume implicitly that the delay requirement $\mathbf{d}_i$ for query Q_i meets two constraints: 1) $\mathbf{d}_i \ge \eta_1 \mathbf{p}_i \mathbf{t}_0$ for some integer constant $\eta_1 \ge 1$; and 2) $\mathbf{d}_i \ge \eta_2 R \mathbf{t}_0$, where $\eta_2 > 0$ is a constant; R is the maximum hop distance from any node in the network to the sink node, which is at least half of the network diameter; and $\mathbf{t}_0$ is the duration of a time-slot. The second condition comes from the fact that, for one-shot query, the minimum delay of answering an aggregation query is $\Omega(R \cdot \mathbf{t}_0)$ (e.g., [19] and [23]) due to the network delay for delivering data from within the network to the sink node. Given a query, the query is said to be satisfied if the query is answered and the delay for the sink to get the answer is finite.

In the example in Fig. 1, assume there is one periodic aggregation query Q_1 with $\chi_1 = 1, \mathbf{p}_1 = 3$. Let $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\} \cup \{v_s\}$. For each of the three sets $\{v_1, v_3, v_7\}, \{v_2, v_4, v_8\}, \{v_5, v_6\}$, all member nodes can transmit concurrently. We give a feasible transmission schedule in Fig. 1 ($k \in \mathbb{N}$). Here, for $i \in \{1, 2, 3, 4, 5\}$, $\{3k+i\}$ means that the corresponding node(s) will transmit at all the time-slots of $3k + i : \forall k \in \mathbb{N}$. By using schedule $\mathcal{S}$ after the arrival of the aggregated result for the first job, all other aggregated results for later jobs arrive in a periodic sequence time of two time-slots apart. Thus, the delay for each period is exactly five time-slots.

Given a set of periodic data aggregation queries $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_m\}$, the second objective is to test whether the set of queries is schedulable.

Definition of Schedulability: Given a network, a set $\mathcal{Q}$ of independent, preemptive, and periodic aggregation queries is *schedulable* if and only if there exists a routing structure and a transmission schedule $\mathcal{S}$ that can answer all queries and the delay of every job $\mathbf{q}_{i,j}$ of each query $Q_i \in \mathcal{Q}$ is *finite*.

If we relax the delay requirement, $\mathcal{Q}$ seems more likely to be satisfied. However, when the network is overloaded with queries

with large request rates, the query set is not schedulable, irrespective of the delay requirement. We will capture the schedulability of a set of queries by *sufficient conditions* and *necessary conditions* with their gap minimized.

Observe that the query in Fig. 1 is schedulable. Given a set of periodic tasks, determining the schedulability has been extensively studied in the real-time literature. Several pioneering results (see [12] and references therein) have been developed for the schedulability test of periodic tasks in single processor. For example, Earliest Deadline First (EDF) can always find a feasible schedule for a set of *schedulable* preemptive, independent, and periodic queries when the delay requirement of each query is at least its period [11].

III. Related Work, Our Preliminaries

In each subsection, in addition to the literature review, we will also present our preliminary result, which serves as a basis for our protocol design.

A. Real-Time Scheduling

Two classes of well-studied real-time scheduling algorithms are rate-monotonic (RM) and EDF scheduling. RM assigns static-priorities to queries on the basis of the cycle duration of the jobs. Liu and Layland [11] presented an RM algorithm in a single processor and the first sufficient condition for schedulability of a set of queries. This is then further extended in [9] and [14]. On the other hand, EDF is a dynamic scheduling algorithm. EDF and its several extensions were proposed to guarantee the end-to-end delay of packets, e.g., EDF with traffic shaper [15]–[17] that can regulate the distorted traffic from the EDF scheduler to deal with the bursty traffic. Unfortunately, using optimal traffic shaper is, in general, infeasible and introduces additional packet delays. Another approach, such as deadline-curve-based EDF (DC-EDF) [26], or similar one [2], is to judiciously adjust the local deadlines of packets at a node, based on the traffic load and/or the end-to-end deadlines. DC-EDF can guarantee end-to-end delay performances and provide a schedulable region as large as that of RC-EDF [26].

Recently, Chipara *et al.* [5], [6] studied the real-time query scheduling in WSNs by assuming a pregiven routing tree, while, in practice, different routing structures have vast impact on the delay performances and flow data rates supported by a WSN.

Next, we present our preliminary result of packet labeling for single-hop queries. Here, a *single-hop query* differs from the query defined in Section II-A in only one aspect: Each packet requests only a single-hop transmission (instead of multihop transmissions across the network). We define the *request rate* of a single-hop query as the reciprocal of its period. Let the utilization of a set of queries be the summation of their request rates.

Definition of Packet Labeling: Given a set $\mathcal{Q}'$ of preemptive and periodic single-hop queries, the objective is to assign a different integer label for each packet, such that if each packet transmits at the time-slot equal to its label, the delay for each query is at most its period.

Observe that a packet labeling scheme corresponds to a single processor periodic job scheduling. Thus, we can label packets based on RM or EDF scheduling.

RM Scheduling: RM prioritizes packets simply based on request rates of queries. When the number of queries is large, RM scheduling can achieve a utilization of 69% (all packets can make their deadlines).

Lemma 1: Given a set $\mathcal{Q}'$ of single-hop queries with utilization at most $\frac{0.69}{N}$, where $N \geq 1$ is an integer constant, there exists a packet labeling scheme such that each label is divided by N (Lemma 1 has found applications in Section IV with the value of N assigned to be either $2c_1c_2$ or $2c_1$; the proof is available in [24]).

EDF Scheduling: EDF prioritizes packets strictly according to their deadlines. EDF can achieve utilization of exactly one [11]. By replacing RM with EDF, we can achieve a result similar to Lemma 1.

B. Min-Delay Aggregation Scheduling

Minimum delay data aggregation problem has been proven to be NP-hard [4], even for the case of simple one-shot query. Authors in [8], [19], [23], and [25] proposed a sequence of constant-ratio approximation algorithms for OQS under PrIM. Based on the related work, we will present our preliminary result of node ranking in a connected dominating set (CDS) (see the definition of CDS in [18]).

Definition of Node Ranking: Given a CDS T_{CDS}, sink v_{s}, interference model, the objective is to assign a rank $r(u)$ for each node u in CDS, such that if all nodes transmit toward the sink v_{s} at the time-slot equal to its rank, the aggregated data from the CDS can be received by v_{s} without interferences.

Clearly, given a CDS, a transmission schedule for OQS with the CDS as input graph corresponds exactly to a node ranking scheme. Moreover, the delay of the schedule corresponds to the maximum rank among all nodes in the CDS. We can compute the ranks of nodes based on existing solutions for OQS. We will focus on a CDS whose maximum node degree is bounded by a constant 12 [19, Lemma 4.1].

Corollary 1: Given a CDS, there exists a node ranking scheme with the maximum rank at most

$$\begin{cases} \beta_\rho(2R+12+O(\log R)), & \text{under PrIM [19]} \\ \beta_2(2R+12+O(\log R)), & \text{under RTS/CTS} \\ (12K^2+1)R+72K^2+O(\log R), & \text{under PhIM [10].} \end{cases}$$

Here, ρ is the interference range under PrIM, β_ρ is a parameter with its value given as $\beta_\rho = \frac{\pi}{\sqrt{3}}\rho^2 + (\frac{\pi}{2}+1)\rho + 1$, R is the graph radius of the CDS, and K is a constant depending on the parameters of PhIM. The value of K is given in [10].

Note that under RTS/CTS, a schedule for OQS with delay $\beta_2(2R + 12 + O(\log R))$ exists due to our following observation: A schedule under PrIM when $\rho = 2$ corresponds to a conflict-free schedule under RTS/CTS for OQS.

We then quantitatively capture the relevance between two nodes' spatial distance and the temporal difference of their ranks. Let $\lambda(\mathcal{M})$ be the *conflict range* of an interference model $\mathcal{M}$ such that for a set of nodes, if the mutual distance between any pair of nodes is at least $\lambda(\mathcal{M})$, then all nodes in this set can transmit concurrently.

Lemma 2 (Spatial-Temporal Relevance): In any node ranking scheme in Corollary 1, for any pair of nodes u, v of mutual dis-

tance at most $\lambda(\mathcal{M})$, $|r(u)-r(v)| < c_1(\mathcal{M})$ with the values of $c_1(\mathcal{M})$ and $\lambda(\mathcal{M})$ given as

$$c_1(\mathcal{M}) = \begin{cases} 2(\rho+1)^2 + O(\log\rho), & \text{under PrIM} \\ 30, & \text{under RTS/CTS} \\ ((K-1)\mathbb{L})^2 + O(\log K\mathbb{L}), & \text{under PhIM} \end{cases}$$

$$\lambda(\mathcal{M}) = \begin{cases} \rho+1, & \text{under PrIM} \\ 2, & \text{under RTS/CTS} \\ (K-1)\mathbb{L}, & \text{under PhIM.} \end{cases}$$

IV. Scheduling Protocol Design

The general framework of our protocol design is universal for various wireless interference models. It consists of three phases.

1) For each query $Q_i \in \mathcal{Q}$, construct a routing tree T_i for data aggregation.
2) For each node, construct a *transmission plan*, which specifies the data to transmit at the current moment. For each query Q_i, based on routing tree T_i, each node u in T_i (u may not be a source node) needs to add data for each period to its plan.
3) For each node u, assign time to transmit for each packet from u's transmission plan. The assignment will rely on our preliminaries of Packet Labeling (Section III-A) and Node Ranking (Section III-B). This phase is the key part.

We then describe each phase in detail.

The first phase is routing. For each aggregation query $Q_i \in \mathcal{Q}$, the routing tree T_i should be a *Steiner Tree* interconnecting the terminals of $\mathcal{S}_i \cup \{v_s\}$. Given a communication graph $G = (V, E)$, we first select a CDS $T_{\rm CDS}$ of G [18]. We then construct a *spanning tree* T_G by connecting each node u not in the CDS to one of u's neighboring dominators. For each query $Q_i \in \mathcal{Q}$, starting with T_G, we prune each node $u \in V$ and an incident communication link $\overrightarrow{uv}$ (from u to its parent node v) in T_G if the intersection of two node sets $\mathcal{S}_i$ and the node set from the subtree of T_G rooted at u (noted as T_G^u) is empty: $\mathcal{S}_i \cap V(T_G^u) = \emptyset$. The pruning operation results in a routing tree T_i for the query Q_i.

The second phase is constructing transmission plans, based on routing trees for aggregation queries. For each query $Q_i \in \mathcal{Q}$ with a routing tree T_i, during each period, first each leaf node in T_i adds the source data to its transmission plan. Then, every internal node in T_i (noted as a *relay* node for query Q_i) only generates one unit of data by aggregating all received data with its own data (if it has), while it may receive multiple data units from its children. Note that, before u adds the data unit to its transmission plan, it needs to wait until receiving the corresponding data from all its children in T_i (the routing tree for query Q_i). Thus, for a query Q_i, the data unit at node u can be either: 1) original, or 2) an aggregated one, depending on whether this data unit comes from one node.

The third phase is packet scheduling at each node that contains data units in its transmission plan. We divide nodes into two complementary groups: nodes not in the CDS $T_{\rm CDS}$ (noted as *leaf nodes*) and nodes in $T_{\rm CDS}$ (noted as *intermediate nodes*). We will ensure that all leaf nodes transmit at even time-slots only, and all intermediate nodes transmit at odd time-slots only; the time-disjoint property can avoid interferences between nodes from different groups.

Packet Scheduling at Leaf Nodes: We employ a grid partition of the deployment plane. The vertical lines $x = i \cdot \lambda$ for $i \in \mathbb{Z}$ and horizontal lines $y = j \cdot \lambda$ for $j \in \mathbb{Z}$ partition the planes into half-open and half-closed grids of side λ (here, λ represents $\lambda(\mathcal{M})$, and $\mathbb{Z}$ represents the integer set)

$$\{[i\cdot\lambda, (i+1)\cdot\lambda \times [j\cdot\lambda, (j+1)\cdot\lambda : i, j \in \mathbb{Z}\}.$$

We then color the grids such that up to one node from every grid with a monotone color can transmit simultaneously. The number of colors used here (noted as $c_2(\mathcal{M})$) depends on the interference model $\mathcal{M}$. Under PrIM, RTS/CTS model, no neighboring grids sharing a common color is enough to avoid interferences, i.e., $c_2(\mathcal{M}) = 4$; while under PhIM, $c_2(\mathcal{M})$ is a larger constant (see [10]). We index the colors used and denote $\sigma_{\rm g}$ as the color of grid $g(\sigma_{\rm g} \in \{0, 1, \ldots, c_2(\mathcal{M})-1\})$.

For each grid g that contains leaf nodes, we find the subset (noted as $V_{\rm leaf}(g)$) of all leaf nodes lying in g. For each node $u \in V_{\rm leaf}(g)$, we find the subset (noted as $\mathcal{Q}_u$) of queries at u (a query $Q_i \in \mathcal{Q}_u$ iff its routing tree contains the node u, i.e., $u \in T_i$). Let $\mathcal{Q}_g = \bigcup_{u\in V_{\rm leaf}(g)} \mathcal{Q}_u$ be the collection of query subsets at leaf nodes from $V_{\rm leaf}(g)$. Note that here, in $\mathcal{Q}_g$, a query Q_i at different nodes is perceived as different queries. Then, we create an instance of Packet Labeling with the input single-hop query set as $\mathcal{Q}_g$. For this instance, when the utilization is larger than the constant $c_3(\mathcal{M})$, the grid g would be overloaded with data transmissions for answering queries in the long run (the complete arguments are available in Section V-C). Thus, no matter what utilizations are for other grids, the query set $\mathcal{Q}$ would be not schedulable due to a "bottleneck" of grid g. On the other hand, when the utilization is smaller than $\frac{0.69}{2c_1c_2}$ (or $\frac{1}{2c_1c_2}$), by Lemma 1, we can obtain a packet labeling scheme by using RM (or EDF) scheduling where each packet (say the pth packet for jth job of query Q_i at some leaf node $u \in V_{\rm leaf}(g)$) is assigned with a label $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle}$ such that $(2c_1c_2) \mid \ell_u^{\langle \mathbf{q}_{i,j},p\rangle}$ (i.e., $2c_1c_2$ divides $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle}$). Here, $c_1(\mathcal{M}), c_2(\mathcal{M})$ are abbreviated to c_1 and c_2.

For every leaf node u, let $\sigma_{\rm g}$ be the color index of the grid g where u lies. We assign the pth packet for query Q_i's jth job at node u with the following transmission time: $t_u^{\langle \mathbf{q}_{i,j},p\rangle} = \ell_u^{\langle \mathbf{q}_{i,j},p\rangle} + 2\sigma_{\rm g}$. This finishes packet scheduling at leaf nodes.

Packet Scheduling at Intermediate Nodes: For each intermediate node u, we find the set of queries $\mathcal{Q}_u$ for which u participates in routing and map each query $Q_i \in \mathcal{Q}_u$ to a new one Q_i' with modification of only the release time: $\mathbf{a}_i' = \mathbf{a}_i + \mathbf{p}_i + 2c_1c_2$. Let $\mathcal{Q}_u' = \bigcup_{Q_i\in\mathcal{Q}_u} Q_i'$. We then create an instance of Packet Labeling with the single-hop query set $\mathcal{Q}_u'$. Note that, for such an instance of Packet Labeling, the utilization is at most $c_1(\mathcal{M}) \cdot \sum_{Q_i\in\mathcal{Q}} \frac{\chi_i}{\mathbf{p}_i}$. If $\sum_{Q_i\in\mathcal{Q}} \frac{\chi_i}{\mathbf{p}_i} > 1$, the sink node would be overloaded with data receptions (the complete arguments are available in Section V-C). Thus, the query set $\mathcal{Q}$ would be not schedulable due to a "bottleneck" of sink node. On the other hand, when $\sum_{Q_i\in\mathcal{Q}} \frac{\chi_i}{\mathbf{p}_i} \le \frac{0.69}{2c_1}$, the utilization of this Packet Labeling instance is at most $\frac{0.69}{2c_1}$, by Lemma 1, we can obtain a packet labeling scheme by using RM (or EDF) scheduling where each packet (say the pth packet for query Q_i's jth job at

TABLE I
PARAMETERS USED IN OUR PROTOCOL DESIGN

T_i	routing tree for query Q_i
ρ	interference range for PrIM
$\mathcal{M}$	interference model *i.e.*,PrIM, CTS/RTS, PhIM
$\lambda(\mathcal{M})$	conflict range
σ_g	color index of the grid g
$\ell_u^{\langle \mathbf{q}_{i,j},p\rangle}$	label of p-th packet for Q_i's j-th job at node u in Packet Labeling
$r(u)$	rank of node u in Node Ranking
$\mathcal{U}_{G,\mathcal{Q}}(g_{v,h})$	initial load of grid $g_{v,h}$
$c_1(\mathcal{M})$ or c_1	refer to Lemma 2
$c_2(\mathcal{M})$ or c_2	number of colors for grid coloring
$c_3(\mathcal{M})$ or c_3	maximum # of nodes transmitting in a grid
$t_u^{\langle \mathbf{q}_{i,j},p\rangle}$	time assigned to p-th packet for Q_i's j-th job

the intermediate node u) is assigned with a label $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle}$ such that $(2c_1) \mid \ell_u^{\langle \mathbf{q}_{i,j},p\rangle}$ (i.e., $2c_1$ divides $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle}$).

For each intermediate node u, for each packet (assume it is the pth packet for Q_i's jth job), we assign a time-slot: $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle} + 2r(u) + 1$. Here, $r(u)$ is the rank of u for Node Ranking with the CDS T_{CDS} as the input. This finishes packet scheduling at intermediate nodes.

To sum up, we present Algorithm 1 for our protocol design and Table I for the notations. We will analyze the performance of Algorithm 1 in Section V-A.

Algorithm 1: Scheduling Protocol for PAQS

Input: A set of periodic aggregation queries $\mathcal{Q}$, an interference model $\mathcal{M}$.

for *each query* $Q_i \in \mathcal{Q}$ **do**
  construct a data aggregation routing tree T_i;
for *each* j*th instance of each query* $Q_i \in \mathcal{Q}$ **do**
  for *each node* u **do**
    if u *is a leaf node in* T_i **then**
      adds the data to u's transmission plan;
    if u *is an internal node in* T_i **then**
      if u *has only one child* v *in* T_i**then**
        when u received the data from v;
        adds data to u's transmission plan;
      else
        when u received data from all children in T_i, generates aggregated data;
        adds the data to u's transmission plan;
for *each leaf node* u (i.e., $u \notin T_{\mathrm{CDS}}$) **do**
  $\sigma_{\mathrm{g}} \leftarrow$ color index of the grid g where u lies;
  for *packet in* u (*pth packet for* $Q_i's$ jth *job*) **do**
    assign time: $t_u^{\langle \mathbf{q}_{i,j},p\rangle} \leftarrow \ell_u^{\langle \mathbf{q}_{i,j},p\rangle} + 2\sigma_{\mathrm{g}}$;
for *each intermediate node* u (i.e., $u \in T_{\mathrm{CDS}}$) **do**
  for *packet at* u (*p*th *packet for* $Q_i's$ jth job) **do**
    assign time: $t_u^{\langle \mathbf{q}_{i,j},p\rangle} \leftarrow \ell_u^{\langle \mathbf{q}_{i,j},p\rangle} + 2r(u) + 1$.
return *Time to transmit for each packet at each node.*

V. SCHEDULABILITY ANALYSIS

In this section, we will propose both sufficient conditions and necessary conditions for schedulability of a set of periodic queries. The validness of proposed sufficient conditions will rely on the correctness of Algorithm 1.

A. Performance of Algorithm 1

All the following results hold whenever the packet label $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle}$ is obtained based on either RM or EDF scheduling. The proofs of Lemmas 3 and 4 and Theorem 1 are available in [24].

Lemma 3: Algorithm 1 can avoid interferences.

Lemma 4: In the schedule output by Algorithm 1, for every job of each query Q_i, for each packet, each leaf node transmits after the release time; each intermediate node transmits after its children in T_i.

Theorem 1: Given a set of periodic aggregation queries $\mathcal{Q}$ and an interference model, for each $Q_i \in \mathcal{Q}$, Algorithm 1 can achieve the end-to-end delays as follows:

$$\begin{cases} 2\beta_\rho(2R + O(\log R)) + 2c_1c_2 + 2\mathbf{p}_i, & \text{under PrIM} \\ 2\beta_2\,(2R + O(\log R)) + 2c_1c_2 + 2\mathbf{p}_i, & \text{RTS/CTS} \\ (12K^2+1)R + O(\log R) + 2c_1c_2 + 2\mathbf{p}_i, & \text{under PhIM} \end{cases}$$

where the parameters ρ, β_ρ, R, K are defined in Corollary 1, $\mathbf{p}_i$ is the period of the query Q_i, c_1 is given in Lemma 2, and c_2 is the number of colors for grid coloring. As it costs at least R time-slots for the sink node to receive the data from the farthest node, combining with Theorem 1, Theorem 2 follows.

Theorem 2 (Approximation Ratio): For each query, assume the end-to-end delay requirement is at least its period (note that this assumption is true for nearly all queries), Algorithm 1 can achieve constant approximation in terms of delay; the approximation ratio is as follows:

$$\begin{cases} \min\left\{4\beta_\rho + \frac{2\mathbf{p}_i + O(\log R)}{R}, 2 + \frac{4\beta_\rho + O(\log R)}{\mathbf{p}_i}\right\}, & \text{under PrIM} \\ \min\left\{4\beta_2 + \frac{2\mathbf{p}_i + O(\log R)}{R}, 2 + \frac{4\beta_2 + O(\log R)}{\mathbf{p}_i}\right\}, & \text{RTS/CTS} \\ \min\left\{c_4 + \frac{2\mathbf{p}_i + O(\log R)}{R}, 2 + \frac{c_4 R + O(\log R)}{\mathbf{p}_i}\right\}, & \text{under PhIM} \end{cases}$$

where the parameters ρ, β_ρ, R, K are defined in Corollary 1, $\mathbf{p}_i$ is the period of the query Q_i, and $c_4 = 12K^2 + 1$.

B. Sufficient Conditions for Schedulable Queries

Given a WSN $G = (V, E)$ and a set of periodic queries $\mathcal{Q}$, we define the *initial utilization* of a node $u \in V$ as $\mathcal{U}_{G,\mathcal{Q}}(u) = \sum_{(u \in \mathcal{S}_i) \wedge (Q_i \in \mathcal{Q})} \frac{\chi_i}{\mathbf{p}_i}$, and the initial utilization of a grid g as the summation of the initial utilizations of all nodes from this grid, i.e., $\mathcal{U}_{G,\mathcal{Q}}(g) = \sum_{u \in V(g)} \mathcal{U}_{G,\mathcal{Q}}(u)$. Note that in the first phase of our protocol design based on RM scheduling, in the instance of Packet Labeling created for each grid g, to generate a packet labeling scheme, the utilization (which is at most $\mathcal{U}_{G,\mathcal{Q}}(g)$) has to be upper-bounded by a constant $\frac{0.69}{2c_1(\mathcal{M})\cdot c_2(\mathcal{M})}$. In addition, in the instance of Packet Labeling created for each intermediate node u, the utilization (which is at most $\sum_{Q_i \in \mathcal{Q}} \frac{\chi_i}{\mathbf{p}_i}$) has to be upper-bounded by a constant $\frac{0.69}{2c_1(\mathcal{M})}$.

Theorem 3: Given a WSN G and a set $\mathcal{Q}$ of periodic queries, Algorithm 1 based on RM scheduling is correct if the following conditions are satisfied:

$$\begin{cases} \mathcal{U}_{G,\mathcal{Q}}(g) & \leq \frac{0.69}{2c_1(\mathcal{M})\cdot c_2(\mathcal{M})} \quad \forall g \\ \sum\limits_{Q_i\in\mathcal{Q}} \frac{\chi_i}{\mathbf{p}_i} & \leq \frac{0.69}{2c_1(\mathcal{M})} \end{cases} \tag{1}$$

where $\mathcal{U}_{G,\mathcal{Q}}(g)$ is the initial utilization of a grid g, $c_1(\mathcal{M})$ is given in Lemma 2, $c_2(\mathcal{M}) = 4$ under PrIM and RTS/CTS model, and $c_2(\mathcal{M}) = K^2$ with K defined in Corollary 1. Similarly, we propose a sufficient condition for the correctness of Algorithm 1 based on EDF scheduling.

Theorem 4: Given a WSN G and a set $\mathcal{Q}$ of periodic queries, Algorithm 1 based on EDF scheduling is correct if the following conditions are satisfied:

$$\begin{cases} \mathcal{U}_{G,\mathcal{Q}}(g) & \leq \frac{1}{2c_1(\mathcal{M})\cdot c_2(\mathcal{M})} \quad \forall g \\ \sum\limits_{Q_i\in\mathcal{Q}} \frac{\chi_i}{\mathbf{p}_i} & \leq \frac{1}{2c_1(\mathcal{M})}. \end{cases} \tag{2}$$

To sum up, we propose a sufficient schedulability test scheme as follows.

Theorem 5 (Sufficient Schedulability Test): Inequality (1) [and Inequality (2)] is a sufficient condition for schedulability of a set of periodic aggregation queries using RM (and EDF respectively) scheduling.

C. Necessary Conditions for Schedulable Queries

Given a set $\mathcal{Q}$ of periodic aggregation queries, for any grid g, if the maximum number of nodes in g that can transmit concurrently is $c_3(\mathcal{M})$, the initial utilization of this grid cannot exceed $c_3(\mathcal{M})$. On the other hand, for a periodic query $Q_i \in \mathcal{Q}$, it costs the sink v_s of time χ_i to receive data during every period $\mathbf{p}_i$, then the initial utilization at sink v_s, given by $\sum_{Q_i\in\mathcal{Q}} \frac{\chi_i}{\mathbf{p}_i}$, cannot exceed one if $\mathcal{Q}$ can be satisfied.

Theorem 6 (Necessary Schedulability Test): If a set of periodic aggregation queries $\mathcal{Q}$ is schedulable under an interference model $\mathcal{M}$, then the following conditions must be satisfied:

$$\begin{cases} \mathcal{U}_{G,\mathcal{Q}}(g) & \leq c_3(\mathcal{M}) \quad \forall g \\ \sum\limits_{Q_i\in\mathcal{Q}} \frac{\chi_i}{\mathbf{p}_i} & \leq 1. \end{cases} \tag{3}$$

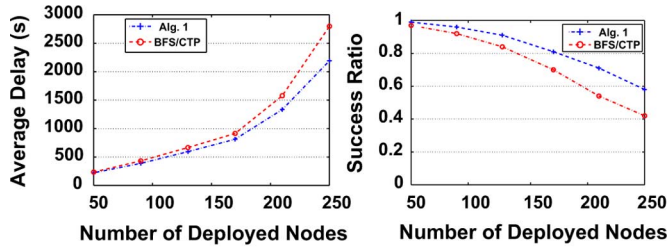
$c_3(\mathcal{M}) \geq 1$ is the maximum number of nodes that can transmit concurrently in a grid under interference model $\mathcal{M}$

$$c_3(\mathcal{M}) = \begin{cases} \frac{16\rho^2}{(\rho-1)^2}, & \text{under PrIM} \\ 36, & \text{under RTS/CTS} \\ \lfloor \frac{2^\kappa \cdot P}{N_0\beta^2} \rfloor, & \text{under PhIM.} \end{cases}$$

If we use RM-based scheduling method instead of EDF-based scheduling, the upper bound of initial utilization for each grid (or node) has to be decreased correspondingly by a factor of 0.69 for schedulable queries.

To conclude, we have the following main theorem.

Theorem 7: The gap between proposed sufficient conditions (Theorem 5) and necessary condition (Theorem 6) for schedulability is $2c_1(\mathcal{M}) \cdot c_2(\mathcal{M}) \cdot c_3(\mathcal{M})$.



(a) (b)

Fig. 2. Increased network size with fixed density. (a) Query delay. (b) Schedulability.

VI. Simulation Results

In this section, we evaluate our scheduling protocol (Algorithm 1) on TOSSIM of TinyOS 2.0.2.

We also implement another scheduling protocol by combining Breadth First Search (BFS) tree and Collection Tree Protocol (CTP, provided in TinyOS 2.0.2) using TOSSIM. The main idea is to construct a BFS tree rooted at the sink node based on the link quality. The original CTP (components) is designed for data collection. To support aggregation, we modified CTP in the upper layer such that each node will not send data to its parent until it aggregates all necessary data from all its children in the BFS tree.

We randomly deploy a number of nodes in a 2-D square region. First, the sink node will broadcast a set of 20 aggregation queries. Each query Q_i has its period, its data type, the required starting time at which each node will start its duty, and IDs of the source nodes. Two data packets can be aggregated into a new data packet iff they are generated at the same period from the same query. The objective is to collect the aggregation ($\max$, $\min$, sum or average) result from all source nodes to the sink node periodically. Once the query procedure begins, the sink node continues to analyze all received data. When all currently existing queries are satisfied, i.e., for each query, the sink node can collect data packets from all source nodes completely and correctly for each period, the sink node will release next 20 queries. Otherwise, the algorithm will terminate. A query will be considered as unsatisfied if the sink node cannot collect complete data packets for this query more than three periods. We use query delay and schedulability as the performance metrics. The schedulability is measured by success ratio, which is the ratio of the number of successful rounds to total rounds for existing queries.

We evaluate under two different network settings. In the first setting, we randomly generated the network topology (connected) with different network size (increasing from 50 to 250) with same network density, i.e., the maximum node degree is fixed to 20. As illustrated in Fig. 2(a), Algorithm 1 outperforms the BFS/CTP method. Fig. 2(b) describes the average success ratio per node with the increment of network size for both methods. Clearly, for both methods, the average success ratio decreases slightly as more nodes are deployed, and Algorithm 1 is better than the BFS/CTP method in most cases.

In the second setting, we fix the deployment area as (300×300) and continue to increase the network size from 50 to 200 with step 30 while keeping the network connected. By

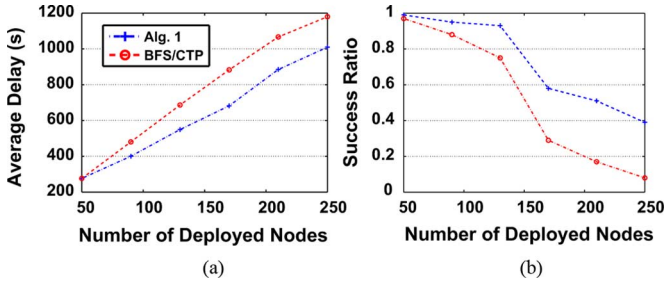


Fig. 3. Increased network size with varied density. (a) Query delay. (b) Schedulability.

doing this, we can fix the radius R and test the performance of both methods with the increment of network density (maximum degree Δ). As we can see from Fig. 3(a), both average delay and success ratio have big gaps between these two methods when network density continues increasing. The reason is that interferences greatly decrease after all the data have been gathered to dominators. For the BFS/CTP method, we continue increasing the number of relay nodes with the increment of network size such that the average delay increases significantly due to the interference. Fig. 3(b) shows the results of success ratio as the network size increases. Algorithm 1 outperforms BFS/CTP method. For Algorithm 1, when the network size increases over 130, the success ratio quickly drops from around 0.9 to 0.6. The BFS/CTP method follows a similar pattern. That is because the new packets from newly increased nodes lead both methods saturated such that many packets are dropped due to the buffer limit.

VII. Discussions

We have proved that our methods can achieve a total rate that is at least a constant factor of the optimum load. There are still a few limitations in our work, which will be summarized as follows. This will serve as some future research challenges.

First, we assumed that a node can aggregate any number of packets into a single packet, while in practice, a different *aggregation degree* may be used, i.e., the size of the aggregated data depends on the number of input data items. One challenge of extending the algorithm to different aggregation degree is to prove its performance. With different aggregation degree, we do not know if the aggregation tree constructed in this paper will still be almost optimal or not. If the tree is still almost optimal, then the link and packet scheduling can be readily extended to address this more challenging case.

Second, we omitted the extra *aggregation delay*. To address this practical challenge, one possible approach is sending a partially aggregated packet without waiting for data packets from all its children nodes. However, this approach may not improve the delay performance; it may hurt it actually. Note that the delay here is defined as the last time the sink collected the data. Although sending a partially aggregated packet allows the sink node to know some partial results earlier, it still did not help the sink to get the data from other children nodes earlier. The sink still has to wait. A potential disadvantage of sending a partially aggregated packet is the increasing of the number of packets to be sent by the node (thus the traffic of the network); this, in turn, will cause the delay onwards because of the additional packets to be sent. A potential advantage of this approach is that, depending on the aggregation function, it may reduce the number of temporarily stored packets at a node (reducing the memory usage, and thus reducing the risk of packets dropping). Therefore, this interesting approach is worthy of extensive future investigation.

In addition, there are some other challenges such as the following: 1) the impact of unreliable network: During data transmissions, sensor nodes and links may suffer from packet losses, which will often trigger rerouting and retransmissions of data. This will incur additional delay and overhead to the network; 2) the impact of the time synchronization errors on the performance of the proposed methods.

VIII. Conclusion

Real-time queries appear in many sensor network applications. For answering periodic queries, we proposed a set of efficient scheduling schemes for data communications. Essentially, we jointly designed the routing strategy as well as packet scheduling protocols under various interference models. Most importantly, we theoretically proved that our algorithm can achieve constant approximation in terms of both delay and schedulability.

Acknowledgment

Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies (NSF and NSFC).

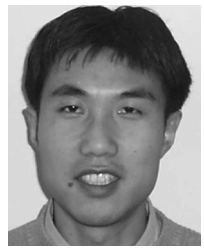
References

[1] M. Alicherry, R. Bhatia, and L. Li, "Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks," in *Proc. ACM MobiCom*, 2005, pp. 58–72.
[2] C. Bouras and A. Sevasti, "A delay-based analytical provisioning model for a QoS-enabled service," in *Proc. IEEE ICC*, 2006, pp. 766–771.
[3] G. Brar, D. Blough, and P. Santi, "Computationally efficient scheduling with the physical interference model for throughput improvement in wireless mesh networks," in *Proc. ACM MobiCom*, 2006, pp. 2–13.
[4] X. Chen, X. Hu, and J. Zhu, "Minimum data aggregation time problem in wireless sensor networks," in *Proc. LNCS*, 2005, pp. 133–142.
[5] O. Chipara, C. Lu, and G. Roman, "Real-time query scheduling for wireless sensor networks," in *Proc. IEEE RTSS*, 2007, pp. 389–399.
[6] O. Chipara, C. Lu, and J. Stankovic, "Dynamic conflict-free query scheduling for wireless sensor networks," in *Proc. IEEE ICNP*, 2006, pp. 321–331.
[7] P. Gupta and P. Kumar, "The capacity of wireless networks," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 388–404, Mar. 2000.
[8] S. Huang, P. Wan, C. Vu, Y. Li, and F. Yao, "Nearly constant approximation for data aggregation scheduling in wireless sensor networks," in *Proc. IEEE INFOCOM*, 2007, pp. 366–372.
[9] J. Leung and J. Whitehead, "Complexity of fixed-priority scheduling of periodic, real-time tasks," *Perform. Eval.*, vol. 2, no. 4, pp. 237–250, 1982.
[10] X.-Y. Li, X. Xu, S. Wang, S. Tang, G. Dai, J. Zhao, and Y. Qi, "Efficient data aggregation in multi-hop wireless sensor networks under physical interference model," in *Proc. IEEE MASS*, 2009, pp. 353–362.
[11] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, 1973.
[12] J. Liu, *Real-Time Systems*. Upper Saddle River, NJ: Prentice Hall, 2000.
[13] L. Mo, Y. He, Y. Liu, J. Zhao, S. Tang, X. Li, and G. Dai, "Canopy closure estimates with GreenOrbs: Sustainable sensing in the forest," in *Proc. ACM SenSys*, 2009, pp. 99–112.

[14] W. Shih, J. Liu, and C. Liu, "Modified rate-monotonic algorithm for scheduling periodic jobs with deferred deadlines," *IEEE Trans. Softw. Eng.*, vol. 19, no. 12, pp. 1171–1179, Dec. 1993.
[15] V. Sivaraman and F. Chiussi, "Providing end-to-end statistical delay guarantees with earliest deadline first scheduling and per-hop traffic shaping," in *Proc. IEEE INFOCOM*, 2000, pp. 631–640.
[16] V. Sivaraman, F. Chiussi, and M. Gerla, "Traffic shaping for end-to-end delay guarantees with EDF scheduling," in *Proc. IEEE IWQoS*, 2000, pp. 10–18.
[17] V. Sivaraman, F. Chiussi, and M. Gerla, "End-to-end statistical delay service under GPS and EDF scheduling: A comparison study," in *Proc. IEEE INFOCOM*, 2001, pp. 1113–1122.
[18] P. A. K. Wan and O. Frieder, "Distributed construction of connected dominating set in wireless ad hoc networks," *Mobile Netw. Appl.*, vol. 9, no. 2, pp. 141–149, 2004.
[19] 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.
[20] J. Wang, Y. Liu, and S. K. Das, "Energy efficient data gathering in wireless sensor networks with asynchronous sampling," *Trans. Sensor Netw.*, vol. 6, no. 3, 2010, Article no. 22.
[21] W. Wang, Y. Wang, X. Li, W. Song, and O. Frieder, "Efficient interference-aware TDMA link scheduling for static wireless networks," in *Proc. ACM MobiCom*, 2006, pp. 262–273.
[22] W. Xi, Y. He, Y. Liu, J. Zhao, L. Mo, Z. Yang, J. Wang, and X. Li, "Locating sensors in the wild: Pursuit of ranging quality," in *Proc. ACM SenSys*, 2010, pp. 295–308.
[23] X. Xu, X. 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.
[24] X. Xu, X.-Y. Li, P.-J. Wan, and S. Tang, "Efficient scheduling for periodic aggregation queries in multihop sensor networks," Illinois Institute of Technology, Chicago, IL, 2011 [Online]. Available: http:cs.iit.edu/~xli/paper/Journal/query-TON2011.pdf
[25] B. Yu, J. Li, and Y. Li, "Distributed data aggregation scheduling in wireless sensor networks," in *Proc. IEEE INFOCOM*, 2009, pp. 2159–2167.
[26] K. Zhu, Y. Zhuang, and Y. Viniotis, "Achieving end-to-end delay bounds by EDF scheduling without traffic shaping," in *Proc. IEEE INFOCOM*, 2001, pp. 1493–1501.

XiaoHua Xu (S'11) received the B.S. degree in computer science from ChuKochen Honors College, Zhejiang University, Hangzhou, China, in 2007, and is currently pursuing the Ph.D. degree in computer science at the Illinois Institute of Technology, Chicago.

His research interests and experience span a wide range of topics from theoretical analysis to practical design in wireless networks.



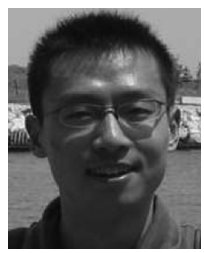
Xiang-Yang Li (SM'08) received the B.Eng. degree in computer science and the Bachelor's degree in business management from Tsinghua University, Beijing, China, in 1995, and the M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana–Champaign in 2000 and 2001, respectively.

He has been an Associate Professor of computer science with the Illinois Institute of Technology, Chicago, since 2006 and was an Assistant Professor from 2000 to 2006. In 2008, he published a monograph, "Wireless Ad Hoc and Sensor Networks: Theory and Applications." His research interests span wireless sensor networks, computational geometry, and algorithms, and he has published over 200 papers on these fields.

Dr. Li is a member of the Association for Computing Machinery (ACM). He is an Editor of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS and *Networks: An International Journal*, and was a Guest Editor of several journals, such as *Mobile Networks and Applications* and the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS.

Peng-Jun Wan received the B.S. degree in applied mathematics from Tsinghua University, Beijing, China, in 1990, the M.S. degree in applied mathematics from the Chinese Academy of Sciences, Beijing, China, in 1993, and the Ph.D. degree in computer and information science from the University of Minnesota, Minneapolis, in 1997.

He is currently a Professor of computer science with the Department of Computer Science, Illinois Institute of Technology, Chicago. His research interests include wireless networks and algorithm design and analysis.



ShaoJie Tang (S'09) received the B.S. degree in radio engineering from Southeast University, Nanjing, China, in 2006, and is currently pursuing the Ph.D. degree in computer science at the Illinois Institute of Technology, Chicago.

His research field is on algorithm design, optimization, security of wireless networks, electronic commerce, as well as online social network.