Efficient Scheduling for Periodic Aggregation Queries in Multihop Sensor Networks

XiaoHua Xu, Shaojie Tang, *Member, IEEE,* XiangYang Li, *Senior Member, IEEE*

Abstract—In this work, 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 $\mathbf{p}_i$, 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 $O(\mathbf{p}_i)$ 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—Query scheduling, aggregation, delay, periodic, interference, utilization, schedulability.

1 Introduction

Wireless sensor networks (WSNs) consists of different types of sensors collaborating to monitor physical or environmental conditions. WSNs are being widely used in cyber-physical systems [24], [37] to provide query services. In response to query requests of a control application, the corresponding sensory data need to be streamed to a control center. In contrast to the direct implementation of raw data collection, in-network aggregation [26], [8], [28] can reduce the requirements for both network bandwidth and power consumptions while guarantee the validities of the aggregated data for answering queries. Thus, for most query services, data aggregation has been a promising approach to reduce energy consumption.

The majority work, *e.g.* [35], [39], [38], [34], [5], [9], [11], [14], [33], for data aggregation scheduling focused on the so-called single *one-shot* query, where each node v_i in the network contains only one data item, say x_i, to be delivered to the control center, and the control center is only interested in getting the aggregated value $f(x_1, x_2, \cdots, x_n)$ 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. For periodic query tasks, sensor nodes are required to deliver their sampled data to the control center for each period. 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 a control center within a delay of seconds so that pursuing actions can be initiated in time. For every period of a query, the delay can be interpreted as the duration from the time when the job for this period is released, to the time when the control center receives all (possibly aggregated) data for this period. Then, queries are often subject to stringent delay constraints, in addition to their already complex appearances (*i.e.*, periodic and multiple queries).

We describe the main problem to be studied as follows. Given a WSN consisting of a set of sensor nodes and a control center (or sink node), the sink node will issue to the network a set $\mathcal{Q}$ of periodic queries for data aggregation. For each query $Q_i \in \mathcal{Q}$, the sink node may be interested in data only from a certain region and therefore, only a subset of sensor nodes

- *The research of authors are partially supported by NSF CNS-0832120, NSF CNS-1035894, program for Zhejiang Provincial Key Innovative Research Team, program for Zhejiang Provincial Overseas High-Level Talents (One-hundred Talents Program), National Basic Research Program of China (973 Program) under grant No. 2010CB328100 and 2010CB334707, and funded by Tsinghua National Laboratory for Information Science and Technology (TNList). Any opinions, findings, conclusions, or recommendations expressed in this paper are those of author(s) and do not necessarily reflect the views of the funding agencies (NSF, and NSFC).*
- *XiangYang Li is with Department of Computer Science, Illinois Institute of Technology, and holds a visiting position at Tsinghua National Laboratory for Information Science and Technology (TNList), and HangZhou DianZi University. Email: xli@cs.iit.edu.*
- *Xiaohua Xu, Shaojie Tang are with Department of Computer Science, Illinois Institute of Technology, Chicago, USA. Emails: {xxu23, stang7}@iit.edu.*

will generate data to satisfy the query; We call these nodes as source nodes. For each period of a query, the sink node expects to receive the corresponding (possibly aggregated) data from the source nodes of this query. For a given wireless interference model (we do not restrict ourself to a specific interference model), the problem **Periodic Aggregation Query Scheduling (PAQS)** seeks to jointly design a routing tree for each aggregation 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).

Numerous milestone results have been developed for various related scheduling problems, *e.g.*, periodic jobs at single processor [17], [15], [41], and packet-level scheduling for the Internet [42], [31]. All the existing results only consider job scheduling at a single node, they did not touch in-network aggregation scheduling for queries, which requires coordinations of data transmissions among all nodes in the WSNs.

On the other hand, when considering aggregation scheduling in the network, due to wireless interferences and resource constraints in WSNs, even for the simple problem of Scheduling for One-shot Query (OQS) which aims to find a schedule to minimize the *delay* for answering this query, it has been proved to be NP-hard [5]. Only a few previous literatures [32], [13], [6] have studied the "real-time" periodic data aggregation/collection scheduling in multi-hop WSNs. Moreover, none of these results provides a theoretical performance assurance on the delay and maximum throughput for WSNs. The main challenges may come from the fact that existing solutions for OQS can not serve directly as a basis for solving the problem **PAQS** after comparing these two problems.

Comparing to One-shot Query Scheduling: While **PAQS** shares the notion of delay bounded in-network aggregation scheduling with the problem OQS, these two problems differ in three aspects. First, the prerequisites for these two problems are different. For OQS, to the best of our knowledge, all methods assume that there are *no* activities scheduled in advance at any node before running the scheduling protocol. However, for multiple periodic queries in **PAQS**, this prerequisite was not satisfied from the second query: when we schedule the i-th query ($i \geq 2$), nodes already have lots of time-slots reserved to serve the first $(i-1)$ queries. In other words, for some nodes, some time-slots are not candidates any more to transmit data for some queries. Note that, we may unify the prerequisites for these two problems by defining a *hyper-period* (the least common multiple of the individual periods). We then merge multiple queries into one meta query with a hyper-period. One main consequence of the unification is the inevitable large delay for some job instances in **PAQS** as we would have many job instances from one query in one hyper-period.

Second, the objectives of two problems are different. For OQS, since each node only needs to transmit once, we can always satisfy this query. The matter here is to minimize the delay of answering the query. However, for periodic queries, we can not take for granted that we can answer them with finite delay. Even if there is only one node in the network (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 towards 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 [18]. 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 the solutions of these two problems are different. For one-shot query, since each node only needs to transmit data once, the solution (schedule) only needs to worry about the interferences from nearby nodes (we call this the intra-query spatial constraints). However, when answering multiple queries, we need to account for additional constraints: (a) for any node, the time-slots scheduled for a period of some query cannot overlap with the time-slots scheduled for another period of the same query or for another query (we call them inter-period node constraints and the inter-query node constraints respectively), (b) 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 (we call this the inter-query spatial constraints and intra-query spatial constraints),

Our Main Contributions: Due to unique challenges for **PAQS**, we need to propose a novel design of scheduling protocols to orchestrate both the real-time job scheduling and in-network aggregation for answering a given set of queries. This is one of the two main contributions of our work.

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. The main idea

for our protocol design is to split the sensor network spatially and temporally and then to find a schedule that makes efficient and careful use of resources. We prove theoretically that our protocol can achieve a total load that is at least a constant fraction of the optimum load; and 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 can test whether a given set of 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 does not satisfy the conditions, we can determine immediately that the WSN 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 of a set of periodic aggregation queries (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 of our protocol design and schedulability test, the result of which corroborate our theoretical analysis.

The rest of the paper is organized as follows. Section 2 formulates the query scheduling problem in WSNs. Section 3 reviews the related work, and introduces our preliminary results. In Section 4, we present our protocol design for scheduling periodic aggregation queries under various interference models. In Section 5, we propose schedulability test schemes for a set of periodic aggregation queries. We present our simulation results in Section 6. Section 7 discusses the limitation of this work and possible future work. Section 8 concludes the paper.

2 System Model, Problem Formulation

2.1 System Model

Let $G = (V, E)$ represent a WSN consisting of a set V of n nodes and a set E of bi-directional 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 frame which 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 time-slot here.

For a set of communication links to transmit simultaneously, we have to avoid wireless 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 [10], 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 [4], [36], 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*, $\texttt{SINR} = \frac{P_u \cdot \|uv\|^{-\kappa}}{\xi + \sum_{w \in I} P_w \cdot \|wv\|^{-\kappa}} \geq \beta$, 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, and $\kappa > 2$ is the path loss exponent, $P_u \in [P_{\min}, P_{\max}]$ is the transmission power of the node u. Here, we assume *uniform power assignment* where each nodes is assigned with the same power, *i.e.*, $P_u = P, \forall u \in V$. Note that under PhIM, to ensure concurrent transmissions, we will construct routing trees in a strong connected communication graph which is a subgraph of G (see [16] for details). In this work, we will extensively study query scheduling and the schedulability test in a WSN under each of these interference models.

Note that for PhIM, we define *maximum transmission radius* as $\mathbb{L} = \sqrt[\kappa]{\frac{P}{\xi\beta}}$, which is the tight upper-bound on the possible communication distances. A node u cannot transmit data to another node v more than the distance $\mathbb{L}$ away even in the absence of other concurrent transmissions. We observe that if a communication link has a length very close to $\mathbb{L}$, the SINR at the corresponding receiver will fluctuate around the threshold β in practice, then the transmission of this link is prone to fail. Thus, to avoid data transmission on the edge of communication boundary, we need to use shorter links, say with length at most $\delta \cdot \mathbb{L}$ with a parameter δ. We can derive a reduced communication graph, denoted as $G_{\delta \cdot \mathbb{L}}(V)$, which consists of all links with length at most $\delta \cdot \mathbb{L}$. If $G_{\delta \cdot \mathbb{L}}(V)$ is connected, we can perform data transmissions in this subgraph of G instead. Therefore, under PhIM, we will construct routing trees in a reduced communication graph $G_{\delta \cdot \mathbb{L}}(V)$.

Assume the control application issues a set of query tasks $\mathcal{Q} = \{Q_1, Q_2, \cdots, Q_m\}$. 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 which 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

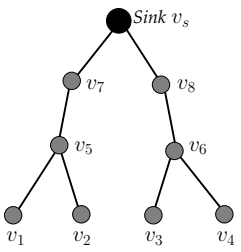
it will take χ_i time to transmit one data unit for query Q_i over any communication link. For different queries, the size of data units may be different; then, χ_i may be different. For Q_i, let $\mathbf{a}_i$ represent the release time (or phase) let $\mathbf{d}_i$ represent the end-to-end *delay requirement* for receiving the answer; then, the j-th 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 towards 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. A packet for temperature cannot simply be aggregated with a packet for humidity for this query. 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 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 $\mathbf{p}_i\mathbf{t}_0$.

2.2 Problem Formulation

Given a set of preemptive, independent and periodic aggregation queries, Here 'independent' means that requests for a certain task do not depend on the initiation or the completion of requests for other tasks [17]. 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 $\{T_i : 1 \leq i \leq m\}$, one tree T_i for each query Q_i. A *transmission schedule*, denoted as $\mathcal{S}$, consists of assigned time-slots for packets at each node to transmit. Let $t_u^{\langle \mathbf{q}_{i,j}, p \rangle}$ be the time-slot assigned to the p-th 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* $\boldsymbol{q}_{i,j}$ 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.*, $D(\mathcal{S}, Q_i, j) = \max_u \max_p \{t_u^{\langle \mathbf{q}_{i,j}, p \rangle} - \big(\mathbf{a}_i + (j-1) \cdot \mathbf{p}_i\big)\}$. 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 query Q_i. We assume implicitly that the delay requirement $\mathbf{d}_i$ for query Q_i meets the following constraints: (1) $\mathbf{d}_i \geq \eta_1 \mathbf{p}_i \mathbf{t}_0$ for some integer constant $\eta_1 \geq 1$, and (2) $\mathbf{d}_i \geq \eta_2 R \mathbf{t}_0$, where $\eta_2 > 0$ is a constant; R is the radius of the network (*i.e.*, 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.*, [38], [34]) 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 membering nodes can transmit cocurrently. 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.

The second objective is the schedulability test.

Definition of Schedulability: Given a network, a set $\mathcal{Q}$ of independent, preemptive, and periodic queries is *schedulable* iff 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*.

Given a set of periodic data aggregation queries $\mathcal{Q} = \{Q_1, Q_2, \cdots, Q_m\}$, the objective is to test whether the set of queries is schedulable. Note that, if we relax the delay requirement, $\mathcal{Q}$ seems to be 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*. In addition, we want to minimize the gap between sufficient conditions and necessary conditions for schedulability of a set of queries.

It is easy to verify that the query given 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 [18] 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 where the delay requirement of each query is at least its period [17].

3 Related Work, Our Preliminaries

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

3.1 Real-time Scheduling

Two representative classes of well-studied real-time scheduling algorithms are rate-monotonic (RM) and EDF scheduling. The class of RM algorithms assigned static-priorities to queries on the basis of the cycle duration of the jobs. Liu and Layland [17] presented a RM algorithm in a single processor, and the first sufficient condition for schedulability of a set of queries. This is then further extended in [15], [27]. 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 [30], [29], [31] 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) [42], or similar one [3], 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 [42].

Recently, Chipara *et al.* [7], [6] studied the real-time query scheduling in WSNs by assuming a pre-given routing tree. Their results often suffer from the fact that 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 2.1 in only one aspect: each packet requests only a single-hop transmission (instead of multi-hop 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: Then, 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 can be divided by N (Lemma 1 has found applications in Section 4 with the value of N assigned to be either $2c_1c_2$ or $2c_1$).

Proof: If each packet would cost N consecutive time-slots instead of one (we assume the period is greater than N here), then the query set $\mathcal{Q}'$ would have a "virtual utilization" of at most 0.69. By using RM scheduling, we can answer all queries and each packet is assigned with N consecutive time-slots. We then select one time-slot among the N time-slots that can be divided by N, and assign it as the packet's label. This finishes our proof. □

EDF Scheduling: We prioritize packets strictly according to their deadlines. EDF can achieve utilization of exactly one instead of 0.69 [17]. By replacing RM with EDF, we can achieve a result similar to Lemma 1.

3.2 Min-Delay Aggregation Scheduling

Minimum delay data aggregation problem has been proven to be NP-hard [5], even for the case of simple one-shot query. Authors in [11], [39], [34] [38] proposed a sequence of constant-ratio approximation algorithms for OQS under PrIM.

Next, we present our preliminary result of node ranking in a Connected Dominating Set (CDS) (see definition in [25]). Let us review the concept of CDS first. In a graph $G = (V, E)$, a subset $V' \subseteq V$ is a

dominating set if every node is either in V' or adjacent to some node in V'. Nodes in V' are called *dominators*; nodes not in V' are called *dominatees*. A subset $V'' \subseteq V$ is a *connected dominating set*, if V'' is a dominating set and V'' induces a connected subgraph.

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 towards 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 will focus on a CDS whose maximum node degree is bounded by a constant 12 (Lemma 4.1 of [34]). Then, we can compute the ranks of nodes based on existing solutions for OQS.

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

$$\begin{cases} \beta_\rho(2R + 12 + O(\log R)) & \text{under PrIM [34]} \\ \beta_2(2R + 12 + O(\log R)) & \text{under RTS/CTS} \\ (12K^2 + 1)R + 72K^2 + O(\log R) & \text{under PhIM [16]} \end{cases}$$

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 [16].

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 distance 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:

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

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

4 Scheduling Protocol Design

The general framework of our protocol design is universal for various wireless interference models.

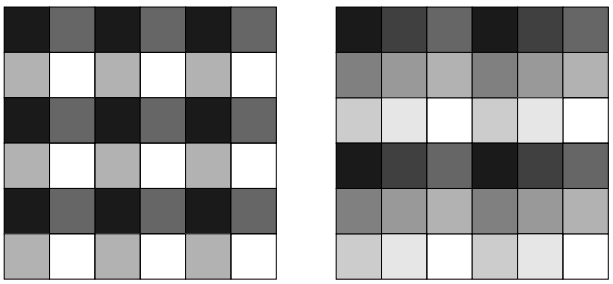
1) For each query $Q_i \in \mathcal{Q}$, construct a routing tree T_i for data aggregation;
2) For each node, construct a *transmission plan*, which specifies the data to transmit at the current moment. For each query Q_i, based on routing tree T_i, each node u in T_i (u may not be a source node) needs to add data for each period to its plan.
3) For each node u, for each packet from u's transmission plan, assign concrete time to transmit. The packet scheduling will be based on our preliminaries of Packet Labeling (Subsection 3.1) and Node Ranking (Subsection 3.2). This phase is the key part.

We then describe the details of each phase separately.

The first phase is routing. The constructions of routing trees are the same under various wireless interference models. Observe that, for each data aggregation query $Q_i \in \mathcal{Q}$, the routing tree T_i should be a *Steiner Tree* inter-connecting the terminals of $\mathcal{S}_i \cup \{v_s\}$. Let the communication graph be $G = (V, E)$, we select a CDS T_{CDS} of G by using an existing approach [25]. We then construct a *spanning tree* T_G by connecting each node u not in the CDS to one of u's neighboring dominators. Based on the spanning tree T_G, for each query $Q_i \in \mathcal{Q}$, we prune each node $u \in V$ and an incident communication link $\overrightarrow{uv}$ (the link 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 data 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) aggregated one, depending on whether this data unit comes from one node.

The third phase is packet scheduling at each node which contains data units in its transmission plan. Here a data unit may consist of several packets. We divide all nodes into two complementary groups: nodes not in the CDS T_{CDS} (noted as *leaf nodes*) and nodes in the CDS T_{CDS} (noted as *intermediate nodes*). We then describe our packet scheduling for each group sepa-



(a) 4-Coloring for PrIM, RTS/CTS

(b) $c_2(\mathcal{M})$-coloring for PhIM

Fig. 2: Grid Partition and Coloring: if at most one node from every grid with a monotone color transmits simultaneously, the transmissions are interference-free. For PhIM, the constant $c_2(\mathcal{M})$ depends on the parameters.

rately. 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 if at most one node from every grid with a monotone color transmits simultaneously, the transmissions are interference-free. Note that, 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 share a common color is enough to avoid interferences, thus, $c_2(\mathcal{M}) = 4$; while under PhIM, $c_2(\mathcal{M})$ is a larger constant (see Fig. 2 and [16]). We index the colors used and denote σ_g as the color of grid g ($\sigma_g \in \{0, 1, \cdots, c_2(\mathcal{M}) - 1\}$).

For each grid g that contains leaf nodes, we find the subset (noted as $V_{\texttt{leaf}}(g)$) of all leaf nodes lying in g; and for each node $u \in V_{\texttt{leaf}}(g)$, 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 $\mathcal{Q}_g = \bigcup_{u \in V_{\texttt{leaf}}(g)} \mathcal{Q}_u$ be the collection of query subsets at leaf nodes from $V_{\texttt{leaf}}(g)$. 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 presented in Subsection 5.3). 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 p-th packet for j-th job of query Q_i at some leaf-node $u \in V_{\texttt{leaf}}(g)$) is assigned with a label $\ell_u^{\langle \mathbf{q}_{i,j}, p \rangle}$; and at the same time, we have $(2c_1c_2) \mid \ell_u^{\langle \mathbf{q}_{i,j}, p \rangle}$ (*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 respectively, we will keep this convention from now on. Note that, for different nodes u and v, the label $\ell_u^{\langle \mathbf{q}_{i,j}, p \rangle}$ should not be equal to $\ell_v^{\langle \mathbf{q}_{i,j}, p \rangle}$, since a packet at two different nodes are perceived as two different packets for labeling.

For every leaf node u, let σ_g be the color index of the grid g where u lies. We assign the p-th packet for query Q_i's j-th job at node u with a transmission time: $t_u^{\langle \mathbf{q}_{i,j}, p \rangle} = \ell_u^{\langle \mathbf{q}_{i,j}, p \rangle} + 2\sigma_g$. 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. Then, we 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 $\mathcal{Q}_u' = \bigcup_{Q_i \in \mathcal{Q}_u} Q_i'$. Finally, we 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, *i.e.*, the sink could not receive all data completely in the long run, for answering the given aggregation query set $\mathcal{Q}$ (The complete arguments are presented in Subsection 5.3). Thus, the query set $\mathcal{Q}$ would be not schedulable, due to a "bottleneck" of the sink node. On the other hand, when $\sum_{Q_i \in \mathcal{Q}} \frac{\chi_i}{\mathbf{p}_i} \leq \frac{0.69}{2c_1}$, 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 p-th packet for query Q_i's j-th job at the intermediate node u)

TABLE 1: 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 number of nodes that can transmit in any grid
$t_u^{\langle \mathbf{q}_{i,j}, p \rangle}$	final scheduled time-slot for p-th packet for Q_i's j-th job

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.,* $u \notin T_{CDS}$*)* **do**
  $\sigma_g \leftarrow$ color index of the grid g where u lies;
  for *packet in* u *(*p*-th packet for* Q_i*'s* j*-th job)* **do**
    assign time: $t_u^{\langle \mathbf{q}_{i,j},p\rangle} \leftarrow \ell_u^{\langle \mathbf{q}_{i,j},p\rangle} + 2\sigma_g$;
for *each intermediate node* u *(i.e.,* $u \in T_{CDS}$*)* **do**
  for *packet at* u *(*p*-th packet for* Q_i*'s* j*-th 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.*

is assigned with a label $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle}$; and at the same time, we have $(2c_1) \mid \ell_u^{\langle \mathbf{q}_{i,j},p\rangle}$ (*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 p-th packet for Q_i's j-th job), we assign it with 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 1 for the notations. We will analyze the performance of Algorithm 1 in Section 5.1.

5 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.

5.1 Performance of Algorithm 1

We first prove the correctness of Algorithm 1. This means that Algorithm 1 outputs an interference-free schedule (Lemma 3) and at the same time, every node will transmit *strictly* after it receives the packets from all its children (Lemma 4), for every period of each query. We then prove the bounded delay of our schedule (Theorem 1). All the results hold whenever the packet label $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle}$ is obtained based on either RM or EDF scheduling in Section 3.1.

Lemma 3: Algorithm 1 can avoid interferences.

Proof: If two packets $p_u^{\langle \mathbf{q}_{i,j},p\rangle}$ and $p_v^{\langle \mathbf{q}_{i',j'},p'\rangle}$ at node u are assigned with the same time-slot, we have either $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle} + 2\sigma_g = \ell_u^{\langle \mathbf{q}_{i',j'},p'\rangle} + 2\sigma_g$ (u is leaf) or $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle} + 2r(u) + 1 = \ell_u^{\langle \mathbf{q}_{i',j'},p'\rangle} + 2r(u) + 1$ (u is intermediate node). Then, $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle} = \ell_u^{\langle \mathbf{q}_{i',j'},p'\rangle}$, this constradicts the fact that labels are different for any single packet labeling instance.

Else, there exist conflicting transmissions. Let u, v be the pair of conflicting nodes with minimum mutual distance and the corresponding transmitting packets be $p_u^{\langle \mathbf{q}_{i,j},p\rangle}$ and $p_v^{\langle \mathbf{q}_{i',j'},p'\rangle}$ respectively. As leaf nodes transmit at even time-slots and intermediate nodes transmit at odd time-slots, u, v must be both leaf nodes or both intermediate nodes.

If u, v are leaf nodes, then packets $p_u^{\langle \mathbf{q}_{i,j},p\rangle}$ and $p_v^{\langle \mathbf{q}_{i',j'},p'\rangle}$ are assigned with $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle} + 2\sigma_g$ and $\ell_v^{\langle \mathbf{q}_{i',j'},p'\rangle} + 2\sigma_{g'}$ respectively. If u, v lie in the same grid, $\sigma_g = \sigma_{g'}$, then $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle} = \ell_v^{\langle \mathbf{q}_{i',j'},p'\rangle}$, contradiction. If u, v lie in different grids g and g', we have $(2c_1c_2) \mid \ell_u^{\langle \mathbf{q}_{i,j},p\rangle}$ ($2c_1c_2$ divides $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle}$) and $(2c_1c_2) \mid \ell_v^{\langle \mathbf{q}_{i',j'},p'\rangle}$. Thus, we have $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle} + 2\sigma_g \equiv \ell_v^{\langle \mathbf{q}_{i',j'},p'\rangle} + 2\sigma_{g'} \mod 2c_1c_2 \Rightarrow 2\sigma_g \equiv 2\sigma_{g'} \mod 2c_1c_2$, then $\sigma_g = \sigma_{g'}$. Thus, g and g' share a common color, *i.e.*, u, v are not conflicting, this leads contradiction.

If u, v are intermediate nodes, then packets $p_u^{\langle \mathbf{q}_{i,j},p\rangle}$ and $p_v^{\langle \mathbf{q}_{i',j'},p'\rangle}$ are assigned with $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle} + 2r(u) + 1$ and $\ell_v^{\langle \mathbf{q}_{i',j'},p'\rangle} + 2r(v) + 1$. We have $2c_1 \mid \ell_u^{\langle \mathbf{q}_{i,j},p\rangle}$ ($2c_1$ divides $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle}$) and $2c_1 \mid \ell_v^{\langle \mathbf{q}_{i',j'},p'\rangle}$. Then, $2r(u) + \ell_u^{\langle \mathbf{q}_{i,j},p\rangle} \equiv 2r(v) + \ell_v^{\langle \mathbf{q}_{i',j'},p'\rangle} \mod 2c_1 \Rightarrow 2r(u) \equiv 2r(v) \mod 2c_1$. If $\|uv\| \le \ell(\mathcal{M})$, by Lemma 2, $|r(u) - r(v)| < c_1 \Rightarrow r(u) = r(v)$; this causes contradiction as $\{r(u) : u \in V(T_{CDS})\}$ is a conflict-free schedule. If $\|uv\| > \ell(\mathcal{M})$, then u, v are not conflicting pair, contradiction. □

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

Proof: Given a packet (say the p-th packet for query Q_i's j-th job), its release time is $\mathbf{a}_i + (j-1)\cdot\mathbf{p}_i$. Clearly, $\mathbf{a}_i + (j-1)\cdot\mathbf{p}_i \le \ell_u^{\langle \mathbf{q}_{i,j},p\rangle} \le \ell_u^{\langle \mathbf{q}_{i,j},p\rangle} + 2\sigma_g = t_u^{\langle \mathbf{q}_{i,j},p\rangle}$.

For any node v, assume node u is a child of v. If u is a leaf node, we have $t_u^{\langle \mathbf{q}_{i,j},p\rangle} = \ell_u^{\langle \mathbf{q}_{i,j},p\rangle} + 2\sigma_g \le \mathbf{a}_i + j\mathbf{p}_i + 2\sigma_g < \mathbf{a}_i + j\mathbf{p}_i + 2c_1c_2$. Here the inequality

$\ell_u^{\langle \mathbf{q}_{i,j},p\rangle} \le \mathbf{a}^i + j\cdot \mathbf{p}_i$ holds because any packet labeling scheme can ensure that each packet is delayed by at most its corresponding period. At the same time, $\ell_v^{\langle \mathbf{q}_{i,j},p\rangle} \ge \mathbf{a}'_i + (j-1)\mathbf{p}_i = (\mathbf{a}_i + \mathbf{p}_i + 2c_1c_2) + (j-1)\mathbf{p}_i = \mathbf{a}_i + j\mathbf{p}_i + 2c_1c_2$. Thus, we have $t_u^{\langle \mathbf{q}_{i,j},p\rangle} < t_v^{\langle \mathbf{q}_{i,j},p\rangle} = \ell_v^{\langle \mathbf{q}_{i,j},p\rangle} + 2r(v) + 1$.

If u is an intermediate node, we want to prove that, for each query, for every packet, node u transmits strictly before its parent v transmits. To begin with, let us construct a packet labeling scheme for all intermediate nodes $\{\ell_u^{\langle \mathbf{q}_{i,j},p\rangle} : \forall i,j,p, \forall u \in T_{CDS}\}$ such that for any intermediate node u, assume v is u's parent in T_G, we have $\forall i,j,p : \ell_u^{\langle \mathbf{q}_{i,j},p\rangle} \le \ell_v^{\langle \mathbf{q}_{i,j},p\rangle}$. We construct labeling of packets at intermediate nodes level by level in tree T_{CDS}, starting from the root v_s. After constructing labeling $\{\ell_v^{\langle \mathbf{q}_{i,j},p\rangle} : \forall i,j,p\}$ for packets at node v. For each child u of v, let $\mathcal{Q}_u$, $\mathcal{Q}_v$ be the query sets for which u, v participates in routing. We observe that $\mathcal{Q}_u \subseteq \mathcal{Q}_v$. Then, for node u, for each packet (say p-th packet for j-th job of query Q_i), we have $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle} = \ell_v^{\langle \mathbf{q}_{i,j},p\rangle}$.

Based on the constructed property of packet labeling, we have $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle} \le \ell_v^{\langle \mathbf{q}_{i,j},p\rangle}$. At the same time, we have $r(u) \le r(v)$ where $r(u) \le r(v)$ are the ranks of node u and v respectively; otherwise the parent v will miss node u's reported data when aggregating data. Then, we have $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle} + 2r(u) + 1 \le + \ell_{p(u)}^{\langle \mathbf{q}_{i,j},p\rangle} + 2r(v) + 1$. By Lemma 3, our schedule can avoid interferences, thus the transmission time of u and v are different for any packet. Thus, we have $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle} + 2r(u) + 1 < \ell_{p(u)}^{\langle \mathbf{q}_{i,j},p\rangle} + 2r(v) + 1$. To sum up, for each query, for every packet, node u transmits strictly before its parent v transmits. This completes the proof. □

Theorem 1: For each query $Q_i \in \mathcal{Q}$, Algorithm 1 can achieve the following end-to-end delays under various wireless interference models:

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

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.

Proof: For the j-th period of Q_i, the end-to-end delay is $\max\{t_u^{\langle \mathbf{q}_{i,j},p\rangle} - \big(\mathbf{a}_i + (j-1)\cdot \mathbf{p}_i\big) : \forall p, \forall u\}$. By Lemma 4, for any packet, intermediate nodes will transmit after leaf nodes transmit, *i.e.*, $\max\{t_u^{\langle \mathbf{q}_{i,j},p\rangle} : \forall p, \forall u \notin T_{CDS}\} \le \max\{t_u^{\langle \mathbf{q}_{i,j},p\rangle} : \forall p, \forall u \in T_{CDS}\}$. We have:

$$\begin{aligned} &\max\{t_u^{\langle \mathbf{q}_{i,j},p\rangle} : \forall p, \forall u\} = \max\{t_u^{\langle \mathbf{q}_{i,j},p\rangle} : \forall p, \forall u \in T_{CDS}\} \\ &= \max\{\ell_u^{\langle \mathbf{q}_{i,j},p\rangle} + 2r(u) + 1 : \forall p, \forall u \in T_{CDS}\} \\ &\le \mathbf{a}'_i + j\cdot \mathbf{p}_i + 2\max\{r(u) : u \in T_{CDS}\} + 1 \\ &= \big(\mathbf{a}_i + \mathbf{p}_i + 2c_1c_2\big) + j\cdot \mathbf{p}_i + 2\max_u\{r(u)\} + 1 \\ &= \ \mathbf{a}_i + (j+1)\cdot \mathbf{p}_i + 2c_1c_2 + 2\max_u\{r(u)\} + 1 \end{aligned}$$

Here, the inequality $\ell_u^{\langle \mathbf{q}_{i,j},p\rangle} \le \mathbf{a}'_i + j\cdot \mathbf{p}_i$ holds because any packet labeling scheme can ensure that each packet is delayed by at most its corresponding period. Thus, the delay is at most

$$2\mathbf{p}_i + 2c_1c_2 + 2\max_u\{r(u)\} + 1.$$

By Corollary 1, $\max_u\{r(u)\} \le$

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

Therefore, the theorem follows. □

As it costs at least R time-slots for 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 $c_4 = 12K^2 + 1$.

5.2 Sufficient Conditions for Schedulable Queries

To characterize a sufficient condition for schedulability, let us first introduce the concept of *initial utilization*. Given a WSN $G = (V, E)$ and a set of queries $\mathcal{Q}$, assume each query $Q_j \in \mathcal{Q}$ is associated with its time χ_j needed by transmitting over a communication link, its period $\mathbf{p}_j$, and a set of source nodes $\mathcal{S}_j \subseteq V$ that contains the needed data. We define the *initial utilization* of a node $u \in V$ as $\mathcal{U}_{G,\mathcal{Q}}(u) = \sum_{(u\in\mathcal{S}_i)\wedge(Q_i\in\mathcal{Q})} \frac{\chi_j}{\mathbf{p}_j}$, 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 (Section 4), for the instance of Packet Labeling created for each grid g, to generate a packet labeling scheme based on RM scheduling, the utilization (which is at most $\mathcal{U}_{G,\mathcal{Q}}(g)$) has to be upper-bounded by a constant

$\frac{0.69}{2c_1(\mathcal{M})\cdot c_2(\mathcal{M})}$. At the same time, for the instance of Packet Labeling created for each intermediate node u, to generate a packet labeling scheme based on RM scheduling, 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 $\frac{0.69}{2c_1(\mathcal{M})}$. To sum up, we propose a sufficient condition for the correctness of Algorithm 1 based on RM scheduling.

***Theorem** 3:* There exists a RM-based conflict-free schedule if the following conditions are satisfied:

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

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 $c_2(\mathcal{M})=K^2$ where the constant K (depending on the parameters of PhIM) is given in [16].

Similar to Theorem 3, we give a sufficient condition for the correctness of Algorithm 1 based on EDF scheduling.

***Theorem** 4:* There exists a EDF-based schedule if the following conditions are satisfied:

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

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.

5.3 Necessary Conditions for Schedulable Queries

Given a set of periodic aggregation queries, to make it possible to schedule nodes' transmissions, observe that, for a subset of nodes where any pair of nodes conflict with each other (the induced conflict graph is a clique), the summation for the initial utilizations of all nodes in this subset can not exceed one. Generally, given a grid g, where the maximum number of nodes in g that can transmit concurrently is a constant $c_3(\mathcal{M})$, the initial utilization of this grid can not exceed $c_3(\mathcal{M})$.

On the other hand, for a single periodic data aggregation query $Q_i\in\mathcal{Q}$, it costs the sink v_s of time χ_i to receive data during every period $\mathbf{p}_i$. Obviously, for a set of queries $\mathcal{Q}$, the initial utilization at sink v_s, given by $\sum_{Q_i\in\mathcal{Q}}\frac{\chi_i}{\mathbf{p}_i}$, can not exceed one if $\mathcal{Q}$ can be satisfied.

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

***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:

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

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

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.

We can obtain the value of $c_3(\mathcal{M})$ without much effort under each wireless interference model as follows:

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

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 a constant $2c_1(\mathcal{M})\cdot c_2(\mathcal{M})\cdot c_3(\mathcal{M})$.

6 Simulation Results

In this section, we present the simulation results which evaluate the proposed algorithm (Algorithm 1). We implemented Algorithm 1 on TOSSIM of TinyOS 2.0.2. In our simulation, we randomly deploy a number of wireless sensor nodes in a 2-D square region, all nodes have the same transmission range. Each node will generate a random 4-byte non-negative number as its own datum. The objective of the sink node is to collect the aggregation ($\max$, $\min$, sum or average) result from all source nodes periodically.

We assume that each node is equipped with 10 types of sensors. Then there are totally 10 types of data in the networks. Two data packets can be aggregated into a new data packet iff they are generated at the same period from the same query. The sink node will generate up to 20 queries with random query parameters. A typical query message Q_i has the following components: the data type, the required starting time at which each node will start its duty, the period, and IDs of the nodes (source nodes) who should respond to the query Q_i.

To evaluate the performance of Algorithm 1, we also implemented another data aggregation method by combining Breath First Search(BFS) tree and CTP (Collection Tree Protocol, which is provided by TinyOS 2.0.2) using TOSSIM. The main idea of BFS/CTP method is to construct a BFS tree rooted at the sink node based on the link quality. In other words, during the procedure of constructing BFS, the

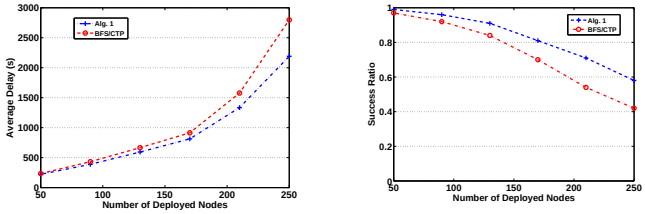
link quality computed by CTP will be considered as the link weight. Notice that, the original CTP method (components) provided in TinyOS 2.0.2 is designed for data collection. To support data aggregation, we modified CTP in the upper layer such that each node will not send data to its parent (in the BFS tree) until it aggregates all necessary data from all its children (in the BFS tree).

The main flow of our evaluation system is as follows. Firstly, the sink node will generate and broadcast a set of data aggregation queries. Secondly, the sink node initiates to construct a CDS (using the algorithms in [25]). Then, a routing tree (based on the CDS) which covers all nodes (may need other relay nodes) will be constructed for data aggregation. When receiving the parameters of start time and time period, each source node generates a datum in each period. All leaf nodes will send data to the corresponding dominators. The dominator will relay data for each period to the sink node based on the routing tree. Basically, each dominator in CDS will locally broadcast information for transmission scheduling along each link to all its dominatees through beacon message.

Once the query procedure begins, the sink node will continue to analyze all received data packets for each period of each query. 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 from all source nodes of this query more than 3 periods.

We use query delay and schedulability as the two metrics to compare the performance of different protocols. The schedulability is measured by success ratio, which is the ratio of the number of successful rounds to total rounds for existing queries. In the following we compare the performance of Algorithm 1 with BFS/CTP under two different network settings. Based on the properties of a wireless TelosB node, each node has a data buffer with size 512 bytes in our simulation. A data packet is 4 bytes including all control and data information. In other words, every node only can temporarily save at most 128 data packets at the same time. When the data buffer is full, the data packet with lowest priority will be dropped. Here, the priority of a data packet is inversely proportional to its period length. If two packets have the same period length, the one with early deadline has higher priority. If the deadlines for two packets are still same, the one received (or generated) firstly has higher priority. It will drop the packet if it already passes deadline.

In the first case, we randomly generated the network topology (connected) with different network size (increasing from 50 to 250) with same network density, *i.e.*, the maximum degree Δ is fixed to 20 in our simulation.

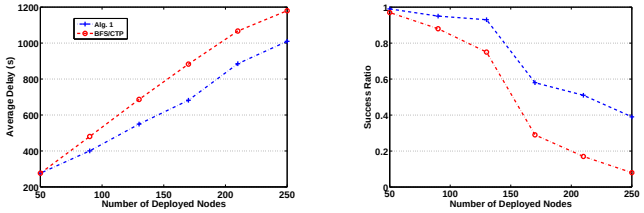


(a) Query Delay (b) Schedulability

Fig. 3: Increased network size with fixed maximum degree.

As illustrated in Fig. 3(a), we can see Algorithm 1 outperforms BFS/CTP. Fig. 3(b) describes the average success ratio per node with the increment of network size for both methods. Clearly, for both algorithms, the average success ratio decrease slightly as the more nodes deployed and our method is better than the other in most cases.

For the second case, 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 is connected. By doing this, we can fix the radius R and test the performance of both algorithms with the increment of network density (maximum degree Δ).



(a) Query Delay (b) Schedulability

Fig. 4: Increased network size with fixed deployment region.

As we can see from Fig. 4(a), both average delay and success ratio have big gaps between these two methods when network density continues increasing. This is mainly due the fact 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. 4(b) shows the results of success ratio as the network size increases. 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. As indicated in the simulation results,

Algorithm 1 has better performance than BFS/CTP method.

7 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 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 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 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 (1) the impact of unreliable network: during data transmissions, sensor nodes and links may suffer from packet losses, which will often trigger re-routing 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.

8 Conclusions

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.

References

[1] Alicherry, M., Bhatia, R., and Li, L. Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks. In *ACM MobiCom* (2005), pp. 58–72.

[2] Beaver, J., Sharaf, M., Labrinidis, A., and Chrysanthis, P. Location-Aware Routing for Data Aggregation in Sensor Networks. In *Geosensor Networks* (2004), pp. 189–209.

[3] Bouras, C., and Sevasti, A. A delay-based analytical provisioning model for a QoS-enabled service. In *IEEE ICC* (2006), pp. 766–771.

[4] Brar, G., Blough, D., and Santi, P. Computationally efficient scheduling with the physical interference model for throughput improvement in wireless mesh networks. In *ACM MobiCom* (2006), pp. 2–13.

[5] Chen, X., Hu, X., and Zhu, J. Minimum Data Aggregation Time Problem in Wireless Sensor Networks. In *LNCS* (2005), pp. 133–142.

[6] Chipara, O., Lu, C., and Roman, G. Real-time query scheduling for wireless sensor networks. In *IEEE RTSS* (2007), pp. 389–399.

[7] Chipara, O., Lu, C., and Stankovic, J. Dynamic conflict-free query scheduling for wireless sensor networks. In *IEEE ICNP* (2006), pp. 321–331.

[8] Considine, J., Li, F., Kollios, G., and Byers, J. Approximate aggregation techniques for sensor databases. In *IEEE ICDE*, pp. 449–460.

[9] Gandham, S., Zhang, Y., and Huang, Q. Distributed Minimal Time Convergecast Scheduling in Wireless Sensor Networks. In *IEEE ICDCS* (2006), pp. 50–50.

[10] Gupta, P., and Kumar, P. The capacity of wireless networks. In *IEEE Transactions on information theory*, (2000), pp. 388–404.

[11] Huang, S., Wan, P., Vu, C., Li, Y., and Yao, F. Nearly Constant Approximation for Data Aggregation Scheduling in Wireless Sensor Networks. In *IEEE INFOCOM* (2007), pp. 366–372.

[12] Intanagonwiwat, C., Estrin, D., Govindan, R., and Heidemann, J. Impact of Network Density on Data Aggregation in Wireless Sensor Networks. In *IEEE ICDCS* (2002), pp. 457–458.

[13] Kalpakis, K., Dasgupta, K., and Namjoshi, P. Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks. In *Computer Networks* (2003), pp. 697–716.

[14] Kesselman, A., and Kowalski, D. Fast distributed algorithm for convergecast in ad hoc geometric radio networks. In *Journal of Parallel and Distributed Computing* (2006), pp. 578–585.

[15] Leung, J., and Whitehead, J. Complexity of fixed-priority scheduling of periodic, real-time tasks. In *Performance Eval* (1982), pp. 237–250.

[16] Li, X.-Y., Xu, X., Wang, S., Tang, S., Dai, G., Zhao, J., and Qi, Y. Efficient Data Aggregation in Multi-hop Wireless Sensor Networks under Physical Interference Model. In *IEEE MASS* (2009), pp. 353–362.

[17] Liu, C. L., and Layland, J. W. Scheduling algorithms for multiprogramming in a hard-real-time environment. In *J. ACM* (1973), pp. 46–61.

[18] Liu, J. In *Real-time systems*. In *Prentice Hall*, (2000).

[19] Luo, H., Liu, Y., and Das, S. K. Routing correlated data with fusion cost in wireless sensor networks. In *IEEE Transactions on Mobile Computing* (2006), pp. 1620–1632.

[20] Luo, H., Liu, Y., and Das, S. K. Distributed algorithm for en route aggregation decision in wireless sensor networks. *IEEE Transactions on Mobile Computing* (2009), pp. 1–13.

[21] Luo, H., Luo, J., Liu, Y., and Das, S. K. Adaptive data fusion for energy efficient routing in wireless sensor networks. In *IEEE Transactions on Computers* (2006), pp. 1286–1299.

[22] LUO, H., TAO, H., MA, H., AND DAS, S. K. Data fusion with desired reliability in wireless sensor networks. In *IEEE Transactions on Parallel and Distributed Systems* (2011), pp. 501–513.

[23] MICHAEL, S., FRANKLIN, M., HELLERSTEIN, J., AND HONG, W. TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks. In *OSDI* (2002).

[24] MO, L., HE, Y., LIU, Y., ZHAO, J., TANG, S., LI, X., AND DAI, G. Canopy closure estimates with greenorbs: Sustainable sensing in the forest. In *ACM SenSys* (2009), pp. 99–112.

[25] P.-J. WAN, K. A., AND FRIEDER, O. Distributed construction of connected dominating set in wireless ad hoc networks. In *IEEE INFOCOM* (2002).

[26] SHARAF, M., BEAVER, J., LABRINIDIS, A., AND CHRYSANTHIS, P. TiNA: a scheme for temporal coherency-aware in-network aggregation. In *Proceedings of the 3rd ACM international workshop on Data engineering for wireless and mobile access* (2003), pp. 69–76.

[27] SHIH, W., LIU, J., AND LIU, C. Modified rate-monotonic algorithm for scheduling periodic jobs with deferred deadlines. *IEEE Transactions on Software Engineering* (1993), pp. 1171–1179.

[28] SHRIVASTAVA, N., BURAGOHAIN, C., AGRAWAL, D., AND SURI, S. Medians and beyond: new aggregation techniques for sensor networks. In *ACM SenSys* (2004), pp. 239–249.

[29] SIVARAMAN, V., AND CHIUSSI, F. Providing end-to-end statistical delay guarantees with earliest deadline first scheduling and per-hop traffic shaping. In *IEEE INFOCOM* (2000).

[30] SIVARAMAN, V., CHIUSSI, F., AND GERLA, M. Traffic shaping for end-to-end delay guarantees with EDF scheduling. In *Proc. of IWQoS* (2000).

[31] SIVARAMAN, V., CHIUSSI, F., AND GERLA, M. End-to-end statistical delay service under GPS and EDF scheduling: A comparison study. In *IEEE INFOCOM* (2001).

[32] TAN, H., AND KÖRPEOGLU, I. Power efficient data gathering and aggregation in wireless sensor networks. *ACM SIGMOD Record* (2003), 66–71.

[33] UPADHYAYULA, S., ANNAMALAI, V., AND GUPTA, S. A low-latency and energy-efficient algorithm for convergecast in wireless sensor networks. In *IEEE GLOBECOM* (2003).

[34] WAN, P.-J., HUANG, S. C.-H., WANG, L., WAN, Z., AND JIA, X. Minimum-latency aggregation scheduling in multihop wireless networks. In *ACM MobiHoc* (2009).

[35] WANG, J., LIU, Y., AND DAS, S. K. Energy efficient data gathering in wireless sensor networks with asynchronous sampling. *ACM Transactions on Sensor Networks* (preprint available online).

[36] WANG, W., WANG, Y., LI, X., SONG, W., AND FRIEDER, O. Efficient interference-aware TDMA link scheduling for static wireless networks. In *Proceedings of the 12th annual international conference on Mobile computing and networking* (2006), ACM, pp. 262–273.

[37] XI, W., HE, Y., LIU, Y., ZHAO, J., MO, L., YANG, Z., WANG, J., AND LI, X. Locating sensors in the wild: pursuit of ranging quality. In *ACM SenSys* (2010), pp. 295–308.

[38] XU, X., LI, X., MAO, X., TANG, S., AND WANG, S. A Delay Efficient Algorithm for Data Aggregation in Multi-hop Wireless Sensor Networks. *IEEE Transactions on Parallel and Distributed Systems* (2011), pp. 163–175.

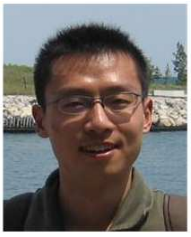
[39] YU, B., LI, J., AND LI, Y. Distributed Data Aggregation Scheduling in Wireless Sensor Networks. In *IEEE INFOCOM* (2009), pp. 2159–2167.

[40] YU, Y., KRISHNAMACHARI, B., AND PRASANNA, V. Energy-latency tradeoffs for data gathering in wireless sensor networks. In *IEEE INFOCOM* (2004).

[41] ZHENG, Q., AND SHIN, K. On the ability of establishing real-time channels in point-to-pointpacket-switched networks. *IEEE Transactions on Communications* (1994), pp. 1096–1105.

[42] ZHU, K., ZHUANG, Y., AND VINIOTIS, Y. Achieving end-to-end delay bounds by EDF scheduling without traffic shaping. In *IEEE INFOCOM* (2001), pp. 1493–1501.

Xiaohua Xu received the BS degree from ChuKochen Honors College at Zhejiang University, P.R. China, in 2007. He is currently working toward the PhD degree in Computer Science at Illinois Institute of Technology. His research interests and experience span a wide range of topics from theoretical analysis to practical design in wireless networks. He is a student member of the IEEE.



Shaojie Tang is a Computer Science PhD student at Illinois Institute of Technology. He received B.S. at Radio Engineering, Southeast University, P.R.China, 2006. His research field is on algorithm design, optimization, security of wireless networks, electronic commerce as well as online social network. He is a student member of IEEE.

Dr. Xiang-Yang Li has been an Associate Professor since 2006 and Assistant Professor of Computer Science at the Illinois Institute of Technology from 2000 to 2006. He hold MS (2000) and PhD (2001) degree at Computer Science from University of Illinois at Urbana-Champaign. He received B.Eng. at Computer Science and Bachelor degree at Business Management from Tsinghua University, P.R. China in 1995. His research interests span wireless sensor networks, computational geometry, and algorithms, and has published over 200 papers on these fields. He is an editor of IEEE TPDS, Networks: An International Journal, and was a guest editor of several journals, such as ACM MONET, IEEE JSAC. In 2008, he published a monograph "Wireless Ad Hoc and Sensor Networks: Theory and Applications". He is a senior member of the IEEE and a member of ACM.