When In-Network Processing Meets Time: Complexity and Effects of Joint Optimization in Wireless Sensor Networks

Qiao Xiang*, Jinhong Xu†, Xiaohui Liu*, Hongwei Zhang*, Loren J. Rittle‡
*Department of Computer Science, Wayne State University, {xiangq27,xiaohui,hongwei}@wayne.edu
†Department of Computer Science, Indiana University, jinhxu@indiana.edu
‡Applied Research and Technology Center, Motorola, ljrittle@motorola.com

***Abstract*—As sensornets are increasingly being deployed in mission-critical applications, it becomes imperative that we consider application QoS requirements in in-network processing (INP). Towards understanding the complexity of joint QoS and INP optimization, we study the problem of jointly optimizing packet packing (i.e., aggregating shorter packets into longer ones) and the timeliness of data delivery. We identify the conditions under which the problem is strong NP-hard, and we find that the problem complexity heavily depends on aggregation constraints (in particular, maximum packet size and re-aggregation tolerance) instead of network and traffic properties. For cases when the problem is NP-hard, we show that there is no polynomial-time approximation scheme (PTAS); for cases when the problem can be solved in polynomial time, we design polynomial time, offline algorithms for finding the optimal packet packing schemes. To understand the impact of joint QoS and INP optimization on sensornet performance, we design a distributed, online protocol *tPack* that schedules packet transmissions to maximize the local utility of packet packing at each node. Using a testbed of 130 TelosB motes, we experimentally evaluate the properties of tPack. We find that jointly optimizing data delivery timeliness and packet packing significantly improve network performance. Our findings shed light on the challenges, benefits, and solutions of joint QoS and INP optimization, and they also suggest open problems for future research.**

***Keywords*-Wireless network, sensor network, real-time, packet packing, in-network processing**

I. Introduction

After the past decade of active research and field trials, wireless sensor networks (which we call *sensornet*s hereafter) have started penetrating into many areas of science, engineering, and our daily life. They are also envisioned to be an integral part of cyber-physical systems such as those for alternative energy, transportation, and healthcare. In supporting mission-critical, real-time, closed loop sensing and control, CPS sensornets represent a significant departure from traditional sensornets which usually focus on open-loop sensing, and it is critical to ensure messaging quality (e.g., timeliness of data delivery) in CPS sensornets. The stringent application requirements in CPS make it necessary to rethink about sensornet design, and one such problem is in-network processing.

For resource constrained sensornets, in-network processing (INP) improves energy efficiency and data delivery performance by reducing network traffic load and thus channel contention. Over the past years, many INP methods have been proposed for query processing (e.g., TinyDB [1]) and general data collection (e.g., DFuse [2]). Not focusing on mission-critical sensornets, however, these works have mostly ignored the timeliness of data delivery when designing INP mechanisms. Recently, Becchetti $et\ al.$ [3] and Oswald $et\ al.$ [4] examined the issue of data delivery latency in in-network processing. Theoretical in nature, these studies assumed *total aggregation* where any arbitrary number of information elements (e.g., reports after an event detection) can be aggregated into one single packet, which may well be infeasible in many practical settings. Thus, the interaction between specific, real-world INP methods and data delivery timeliness remains a largely unexplored issue in sensornet systems. This is an important issue because 1) it affects the efficiency and quality of real-time embedded sensing and control, and 2) as we will show later in the paper, data aggregation constraints (e.g., aggregation capacity limit and re-aggregation tolerance) affect, to a greater extent than network and traffic properties, the complexity and the protocol design in jointly optimizing INP and the timeliness of data delivery.

Towards understanding the interaction between INP and data delivery latency in foreseeable real-world sensornet deployments, we focus on a widely used, application-independent INP method — *packet packing* where multiple short packets are aggregated into a single long packet [5], [6]. In sensornets (especially those for real-time sensing and control), an information element from each sensor is usually short, for instance, less than 10 bytes [7], [1]. Yet the header overhead of each packet is relatively high in most sensornet platforms, for instance, up to 31 bytes at the MAC layer alone in IEEE 802.15.4 based networks. It is also expected that more header overhead will be introduced at other layers (e.g., routing layer) as we standardize sensornet protocols such as in the effort of the IETF working groups 6LowPAN [8] and ROLL [9]. Besides header overhead, MAC coordination also introduces non-negligible overhead in wireless networks [6]. If we only transmit one short information element in each packet transmission, the high overhead in packet transmission will significantly reduce the network throughput; this is especially the case for high speed wireless networks such as IEEE 802.15.4a ultrawideband (UWB) networks. Fortunately, the maximum size of packet payload is usually much longer than that of each information element, for instance, 128 bytes per MAC frame in 802.15.4. Therefore, we can aggregate multiple information elements into a single packet to reduce the amortized overhead of transmitting each element. Packet packing also reduces the

number of packets contending for channel access, hence it reduces the probability of packet collision and improves information delivery reliability, as we will show in Section V. The benefits of packet packing have also been recognized by the IETF working groups 6LowPAN and ROLL.

Unlike total aggregation assumed in [3] and [4], the number of information elements that can be aggregated into a single packet is constrained by the maximum packet size, thus we have to carefully schedule information element transmissions so that the degree of packet packing (i.e., the amount of sensing data contained in packets) can be maximized without violating application requirement on the timeliness of data delivery. As a first step toward understanding the complexity of jointly optimizing INP and QoS with aggregation constraints, we analyze the impact that aggregation constraints have on the computational complexity of the problem, and we prove the following:

- When a packet can aggregate three or more information elements, the problem is strong NP-hard, and there is no polynomial-time approximation scheme (PTAS).
- When a packet can only aggregate two information elements, the complexity depends on whether two elements in a packet can be separated and re-packed with other elements on their way to the sink: if the elements in a packet can be separated, the problem is strong NP-hard and there is no PTAS for the problem; otherwise it can be solved in polynomial time by modeling the problem as a maximum weighted matching problem in an interval graph.
- The above conclusions hold whether or not the routing structure is a tree or a linear chain, and whether or not the information elements are of equal length.

Besides shedding light on the complexity and protocol design of jointly optimizing data delivery timeliness and packet packing (as well as other INP methods), these findings incidentally answer several open questions on the complexity of batch-process scheduling in interval graphs [10].

To understand the impact of jointly optimizing packet packing and data delivery timeliness, we design a distributed, online protocol *tPack* that schedules packet transmissions to maximize the local utility of packet packing at each node. Using a testbed of 130 TelosB motes, we experimentally evaluate the properties of tPack. We find that jointly optimizing data delivery timeliness and packet packing significantly improve network performance (e.g. in terms of high reliability, high energy efficiency, and low delay jitter).

The rest of the paper is organized as follows. We discuss the system model and precisely define the joint optimization problem in Section II. Then we analyze the complexity of the problem in Section III, and present the tPack protocol in Section IV. We experimentally evaluate the performance of tPack and study the impact of packet packing as well as joint optimization in Section V. We discuss related work in Section VI, and conclude the paper in Section VII. *Due to the limitation of space, we relegate the proofs of most of the theorems of this paper to [11].*

II. System model and problem definition

A. System model

We consider a directed collection tree $T = (V, E)$, where V and E are the set of nodes and edges in the tree. $V = \{v_i : i = 1 \dots N\} \cup \{R\}$ where R is the root of the tree and represents the data sink of a sensornet, and $\{v_i : i = 1 \dots N\}$ are the set of N sensor nodes in the network. An edge $\langle v_i, v_j \rangle \in E$ if v_j is the parent of v_i in the collection tree. The parent of a node v_i in T is denoted as p_i. We use $ETX_{v_i v_j}(l)$ to denote the expected number of transmissions required for delivering a packet of length l from a node v_i to its ancestor v_j, and we use $t_{v_i v_j}(l)$ to denote the time taken to deliver a packet of length l from v_i to v_j.

Each information element x generated in the tree is identified by a 4-tuple (v_x, l_x, r_x, d_x) where v_x is the node that generates x, l_x is the length of x, r_x is the time when x is generated, and d_x is the deadline by which x needs to be delivered to the sink node R. We use $s_x = d_x - (r_x + t_{v_x R}(l_x))$ to denote the *spare time* for x, and we define the *lifetime of* x as $[r_x, d_x]$.

B. Problem definition

Given a collection tree T and a set of information elements $X = \{x\}$ generated in the tree, we define the problem of jointly optimizing packet packing and the timeliness of data delivery as follows:

Problem $\mathbb{P}$: given T and X, schedule the transmission of each element in X to minimize the total number of packet transmissions required for delivering X to the sink R while ensuring that each element be delivered to R before its deadline.

In an application-specific sensornet, the information elements generated by different nodes depend on the application but may well be of equal length [7]. Depending on whether the sensornet is designed for event detection or data collection, moreover, the information elements X may follow certain arrival processes. Based on the specific arrival process of X, the following special cases of problem $\mathbb{P}$ tend to be of practical relevance in particular:

Problem $\mathbb{P}_0$: same as $\mathbb{P}$ except that 1) the elements of X are of equal length, and 2) X includes at most one element from each node; this problem can represent sensornets that detect rare events.

Problem $\mathbb{P}_1$: same as $\mathbb{P}$ except that 1) the elements of X are of equal length, and 2) every two consecutive elements generated by the same node v_i are separated by a time interval whose length is randomly distributed in $[a, b]$; this problem can represent periodic data collection sensornets (with possible random perturbation to the period).

Problem $\mathbb{P}_2$: same as $\mathbb{P}$ except that the elements of X are of equal length; this problem represents general application-specific sensornets.

III. Complexity of joint optimization

The complexity of problem $\mathbb{P}$ depends on aggregation constraints such as maximum packet size and whether infor-

mation elements in a packet can be separated and repacked with other elements. For convenience, we use K to denote the maximum number of information elements that can be packed into a single packet. (Note that K depends on the maximum packet size and the lengths of information elements in problem $\mathbb{P}$.) In what follows, we first analyze the case when $K \geq 3$ and then the case when $K = 2$, and we discuss how aggregation constraints affect the problem complexity.

A. Complexity when $K \geq 3$

We first analyze the complexity and the hardness of approximation for problem $\mathbb{P}_0$, then we derive the complexity of $\mathbb{P}_1$, $\mathbb{P}_2$, and $\mathbb{P}$ accordingly. The analysis is based on reducing the Boolean-satisfiability-problem (SAT) [12] to $\mathbb{P}_0$ as we show below.

Theorem 1: When $K \geq 3$, problem $\mathbb{P}_0$ is strong NP-hard whether or not the routing structure is a tree or a linear chain.

Proof: To prove that $\mathbb{P}_0$ is strong NP-hard, we first present a polynomial transformation f from the SAT problem to $\mathbb{P}_0$, then we prove that an instance Π of SAT is satisfiable if and only if the optimal solution of $\Pi' = f(\Pi)$ has certain minimum number of transmissions.

Given an instance Π of the SAT problem which has n Boolean variables $X_1, \ldots, X_n$ and m clauses $C_1, \ldots, C_m$, we derive a polynomial time transformation from Π to an instance Π' of $\mathbb{P}_0$ with $K \geq 3$ as follows. We first construct a tree as shown in Figure 1. In this tree, node v_j, $j = 1, \ldots, n$, corresponds to the variable X_j. Node v is an intermediate node, and node S is the sink node. $ETX_{v_j v}$ is D with $D \gg 1$, and ETX_{vs} is 1. If a variable X_j appears k_j times in total in the m clauses, then $2k_j + 3$ children nodes are attached to node v_j, labeled as $v_0^j, \ldots, v_{2k_j+2}^j$. m children are also attached to node v, labeled as $v_1^c, \ldots, v_m^c$. Each of these edge has a ETX of 1. The transmission time from each child of v_j to itself is t_1, and the transmission time from v_i^c to v is t_4.

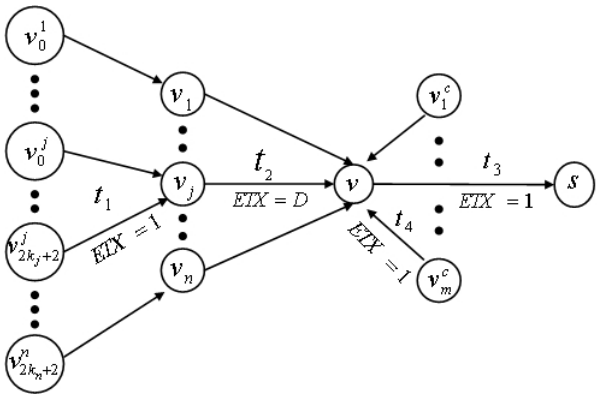

Figure 1. Reduction from SAT to $\mathbb{P}_0$ when $K \geq 3$

After constructing the tree, we define the information elements and their lifetimes as follows. For each subtree rooted at node v_j, we first define $2k_j + 1$ information elements and then assign them one by one to the leaf nodes $v_1^j, \ldots, v_{2k_j+1}^j$ of this subtree. If variable X_j occurs unnegated in clause C_i, we create an information element x_i^j with lifetime $[r_i^j, d_i^j] = [(3i+1)(n+1)+j, (3i+2)(n+1)+j+t_1+t_2+t_3]$. If X_j occurs negated in clause C_i, we create an information element $x_i^j : [r_i^j, d_i^j] = [3i(n+1)+j, (3i+1)(n+1)+j+t_1+t_2+t_3]$. Let $i_1^j < \ldots < i_{k_j}^j$ denote the indices of the clauses in which variable X_j occurs. For every two messages $x_{i_t^j}^j$ and $x_{i_{t+1}^j}^j$, $t = 1, \ldots, k_j - 1$, define an information element $ax_{i_t^j}^j : [r_{a_t}^j, d_{a_t}^j] = [d_{i_t^j}^j - t_1 - t_2 - t_3, r_{i_{t+1}^j}^j + t_1 + t_2 + t_3]$. We also define $ax_0^j : [r_{a_0}^j, d_{a_0}^j] = [j, r_{i_1^j}^j + t_1 + t_2 + t_3]$, and $ax_{k_j}^j : [r_{a_{k_j}}^j, d_{a_{k_j}}^j] = [d_{i_{k_j}^j}^j - t_1 - t_2 - t_3, 3(m+1)(n+1)+j+t_1+t_2+t_3]$. In this way, every two consecutive information elements in this sequence overlap in their lifetimes, and the size of the overlap is $t_1 + t_2 + t_3$. After defining these $2k_j + 1$ information elements, we set the source of each element one by one from node v_1^j to node $v_{2k_j+1}^j$. For each node v_0^j, we define an element $z_0^j : [j, j + t_1 + t_2 + t_3]$. For each node $v_{2k_j+2}^j$, we define an element $z_{2k_j+2}^j : [3(m+1)(n+1)+j, 3(m+1)(n+1)+j+t_1+t_2+t_3]$. Figure 2 demonstrates how the lifetimes of these $2k_j + 3$ information elements are defined.

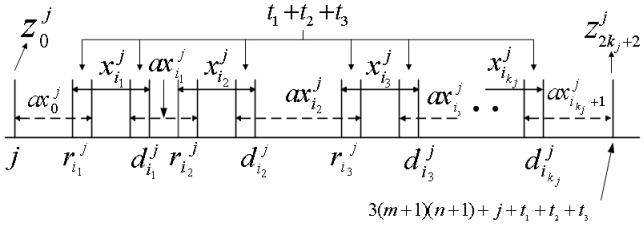

Figure 2. Lifetimes of information elements

Similarly, we define m information elements generated by nodes $v_1^c, \ldots, v_m^c$, with element $z_i : [r_i, d_i] = [(3i+1)(n+1)+t_1+t_2-t_4, (3i+2)(n+1)+t_1+t_2+t_3]$, $i = 1, \ldots, m$, being generated by node v_i^c. Then, for nodes v_1 to v_n, we define an information element for each of them with lifetime $[4(m+1)(n+1)+i, 4(m+1)(n+1)+i+t_2+t_3]$, $i = 1, \ldots, n$. For node v, define an information element with lifetime $[4(m+1)^2(n+1)+i, 4(m+1)^2(n+1)+i+t_3]$.

Given the above polynomial-time reduction from a SAT problem Π to an instance Π' of $\mathbb{P}_0$, we prove that the minimum number of transmissions required in Π' is $\sum_{j=1}^{n}(2k_j+1)+\sum_{j=1}^{n}[(k_j+1)(D+1)]+2n(D+1)+2n+m+1$ if and only if Π is satisfiable. We also find that the above reduction can be extended to the case when the routing structure is a linear chain in $\mathbb{P}_0$. Due to the limitation of space, we relegate the detailed discussion to [11]. Therefore, $\mathbb{P}_0$ is strong NP-hard when $K \geq 3$, whether or not the routing structure is a tree or a linear chain. ■

Having proved the strong NP-hardness of $\mathbb{P}_0$ when $K \geq 3$, we analyze the hardness of approximation for $\mathbb{P}_0$ using a gap-preserving reduction from MAX-3SAT to $\mathbb{P}_0$ [13], and

we have

Theorem 2: When $K \geq 3$, there exists $\epsilon \geq 1$ such that it is NP-hard to achieve an approximation ratio of $1+\frac{1}{200N}(1-\frac{1}{\epsilon})$ for problem $\mathbb{P}_0$, where N is the number of information elements in $\mathbb{P}_0$.

Proof: Interested readers can find the proof in [11]. ■

Based on the definition of polynomial time approximation scheme (PTAS) and Theorem 2, we then have

Corollary 1: There is no polynomial time approximation scheme (PTAS) for problem $\mathbb{P}_0$ when $K \geq 3$.

Based on the findings for $\mathbb{P}_0$, we have

Theorem 3: When $K \geq 3$, problems $\mathbb{P}_1$, $\mathbb{P}_2$, and $\mathbb{P}$ are strong NP-hard whether or not the routing structure is a tree or a linear chain, and there is no polynomial-time approximation scheme (PTAS) for solving them.

Proof: Interested readers can find the proof in [11]. ■

Theorems 1 and 3 show that the joint optimization problems are strong NP-hard and there is no PTAS, whether or not the routing structure is a tree or a linear chain and whether or not the information elements are of equal length. In contrast, Becchetti $et\ al.$ [3] showed that, for total aggregation, the joint optimization problems are solvable in polynomial time via dynamic programming on chain networks. Therefore, we see that aggregation constraints make the difference on whether a problem is tractable for certain networks, and thus it is important to consider them in the joint optimization. Incidentally, we note that Theorem 3 also answers the open question on the complexity of Problem (P4) of batch-process scheduling in interval graphs [10].

B. Complexity when $K = 2$

We showed in Section III-A that the problems $\mathbb{P}_i, i = 0, 1, 2$, and $\mathbb{P}$ are all strong NP-hard and there is no PTAS for these problems when $K \geq 3$. We prove in this section that, when $K = 2$, the complexity of these problems depends on whether information elements in a packet can be separated and re-packed with other elements (which we call *re-aggregation* hereafter) on their way to the sink. When re-aggregation is disallowed, these problems are solvable in polynomial time; otherwise they are strong NP-hard. Note that, when $K \geq 3$, these problems are all strong NP-hard even if re-aggregation is disallowed, which can be seen from the proof of Theorem 1. Note also that, even though re-aggregation may well be allowed in most sensornet systems when the in-network processing (INP) method is packet packing, re-aggregation may not be possible or allowed when INP is data fusion such as lossy data compression [14]. Via the study on the impact of re-aggregation, therefore, we hope to shed light on the structure of the joint optimization problems when general INP methods are considered.

When $K = 2$ and re-aggregation is allowed, the complexity of the joint optimization problems is very much similar to the case when $K \geq 3$; that is, these problems are all strong NP-hard and there is no PTAS, whether or not the routing structure is a tree or a linear chain and whether or not the information elements are of equal length. Due to the limitation of space, we relegate the detailed proofs to [11], and we only discuss in detail the case when re-aggregation is prohibited as follows.

When $K = 2$ and re-aggregation is prohibited, we can solve problem $\mathbb{P}$ (and thus its special versions $\mathbb{P}_0$, $\mathbb{P}_1$, and $\mathbb{P}_2$) in polynomial time by transforming it into a maximum weighted matching problem in an interval graph. An interval graph G_I is a graph defined on a set I of intervals on the real line such that 1) G_I has one and only one vertex for each interval in the set, and 2) there is an edge between two vertices if the corresponding intervals intersect with each other. Given an instance of problem $\mathbb{P}$, we solve it using Algorithm 1 as follows:

Algorithm 1 Algorithm for solving $\mathbb{P}$ when $K = 2$ and re-aggregation is prohibited

1: Generate an interval graph $G_I(V_I, E_I)$ for problem $\mathbb{P}$ as follows:
- Select an arbitrary information element q generated by node v_q at time r_q and with spare time s_q, define an interval $[r_q, r_q + s_q]$ for q on the real line.
- For each remaining information element p generated by node v_p at time r_p and with spare time s_p, let node v_{pq} be the common ancestor of v_p and v_q that is the farthest away from R among all common ancestors of v_p and v_q, then define an interval $[r_q - t_{v_q v_{pq}} + t_{v_p v_{pq}}, r_q - t_{v_q v_{pq}} + t_{v_p v_{pq}} + s_q]$ for information element p.
- Let $V_I = \emptyset$. Then, for each information element s, define a vertex s and add it to V_I.
- Let $E_I = \emptyset$. If the two intervals that represent any two information elements u and h overlap with each other, define an edge (u, h) and add it to E_I; then assign edge (u, h) with a weight $com(u, h) = ETX_{v_{uh}R}(l_u) + ETX_{v_{uh}R}(l_h) - ETX_{v_{uh}R}(l_u + l_h)$, where l_u and l_h are the length of u and h respectively.

2: Solve the maximum weighted matching problem for G_I using Edmonds' Algorithm [15].

3: For each edge (u, h) in the matching, information elements u and h are packed together at node v_{uv}. For all other vertices not in the matching, their corresponding information elements are sent to the sink alone without being packed with any other information element.

For Algorithm 1, we have

Theorem 4: When $K = 2$ and re-aggregation is prohibited, Algorithm 1 solves problem $\mathbb{P}$ in $O(n^3)$ time, where n is the number of information elements considered in the problem.This holds whether or not the routing structure is a tree or a linear chain, and whether or not the information elements are of equal length.

Proof: It is easy to see that if information elements u and h are packed together, the total number of transmissions taken to deliver u and h is $ETX_{v_u R}(l_u) + ETX_{v_h R}(l_h) - ETX_{v_{uh}R}(l_u) - ETX_{v_{uh}R}(l_h) + ETX_{v_{uh}R}(l_u + l_h) = ETX_{v_u R}(l_u) + ETX_{v_h R}(l_h) - com(u, h)$. Let V_I be the

set of vertices in the interval graph G_I, M be a matching in G_I, V_1 be the set of nodes in M, and $V_2 = V_I/V_1$. Then the weight of M, denoted by W_M, is as follows:

$$\begin{aligned} W_M &= \sum_{(u,h)\in M} com(u,h) \\ &= \sum_{(u,h)\in M}[ETX_{v_uR}(l_u) + ETX_{v_hR}(l_h) \\ &\quad -(ETX_{v_uR}(l_u) + ETX_{v_hR}(l_h) \\ &\quad -com(u,h))] \\ &= \sum_{(u,h)\in M}(ETX_{v_uR}(l_u) + ETX_{v_hR}(l_h)) \\ &\quad -\sum_{(u,h)\in M}[ETX_{v_uR}(l_u) + ETX_{v_hR}(l_h) \\ &\quad -com(u,h)] \\ &= \sum_{s\in V_1} ETX_{sR}(l_s) + \sum_{v\in V_2} ETX_{vR}(l_v) \\ &\quad -\{\sum_{(u,h)\in M}[ETX_{v_uR}(l_u) + ETX_{v_hR}(l_h) \\ &\quad -com(u,h)] \\ &\quad +\sum_{v\in V_2} ETX_{vR}(l_v)\} \\ &= \sum_{v\in V_I} ETX_{vR}(l_v) \\ &\quad -\{\sum_{(u,h)\in M}[ETX_{v_uR}(l_u) + ETX_{v_hR}(l_h) \\ &\quad -com(u,h)] + \sum_{v\in V_2} ETX_{vR}(l_v)\} \end{aligned} \tag{1}$$

Note that $\sum_{v\in V_I} ETX_{vR}(l_v)$ is a fixed value, and $\sum_{(u,h)\in M}[ETX_{v_uR}(l_u) + ETX_{v_hR}(l_h) - com(u,h)] + \sum_{v\in V_2} ETX_{vR}(l_v)$ is the total number of transmissions, denoted by ETX_{total}, incurred in the packing scheme generated by Algorithm 1. Therefore, ETX_{total} is minimized if and only if W_M is maximized, which means that solving the maximum weighted matching problem can give us an optimal solution to the original packet packing problem.

Let n denote the total number of information elements in this problem. The whole algorithm consists of three parts. The first one is to define an interval graph and assign weights to each node and edge in the graph, whose time complexity is $O(n^2)$. The second part is to solve the maximum weighted matching problem, whose time complexity is $O(n^3)$ by Edmonds' Algorithm [15]. And the third part is to convert the optimal matching problem to the optimal packing scheme, whose time complexity is $O(n)$. Therefore, the time complexity of the whole algorithm is $O(n^2) + O(n^3) + O(n) = O(n^3)$.

■

By the definition of the weight $com(u,h)$ for elements u and h in Algorithm 1, the solution generated by the maximum weighted matching tends to greedily pack elements as soon as possible after they are generated. This observation motivates us to design a local, greedy online algorithm ***tPack*** in Section IV for the general joint optimization problems, and the effectiveness of this approach will be demonstrated through competitive analysis and testbed based measurement study in Sections IV and V. Note that, incidentally, Theorem 4 also answers the open question on the complexity of scheduling batch-processes with release times in interval graphs [10].

IV. A UTILITY BASED ONLINE ALGORITHM

We see from Section III that problem $\mathbb{P}$ and its special cases in sensornets are strong NP-hard in most system settings, and there is no polynomial-time approximation scheme (PTAS) for these problems. Instead of trying to find global optimal solution, therefore, we focus on designing a local, distributed online algorithm ***tPack*** that optimizes the local utility of packet packing at each node.

Based on the definition of $\mathbb{P}$, its optimization objective is to minimize

$$AC = \frac{TX_{net}}{data_{net}} \tag{2}$$

where TX_{net} is the total number of transmissions taken to deliver $data_{net}$ amount of data to the sink before their deadlines. For convenience, we call AC the *amortized cost* of delivering $data_{net}$ amount of data. In what follows, we design an online algorithm tPack based on this concept of amortized cost of data transmission.

When node j has a packet pkt in its data buffer, j can decide to transmit pkt immediately or to hold it. If j transmits pkt immediately, information elements carried in pkt may be packed with packets at j's ancestors to reduce the amortized cost of data transmissions from those nodes; if j holds pkt, more information elements may be packed with pkt so that the amortized cost of transmission from j can be reduced. Therefore, we can define the *utility* of transmitting or holding pkt as the expected reduction in amortized data transmission cost as a result of the corresponding action, and then the decision on whether to transmit or to hold pkt depends on the utilities of the two actions. For simplicity and for low control overhead, we only consider the immediate parent of node j when computing the utility of transmitting pkt. We will show the goodness of this local approach through competitive analysis later in this section and through testbed based measurement in Section V.

In what follows, we first derive the utilities of holding and transmitting a packet, then we present a scheduling rule that improves the overall utility.

A. Utility calculation

For convenience, we define the following notations:

L	:	maximum payload length per packet;
$ETX_{jp}(l)$	:	expected number of transmissions taken to transport a packet of length l from node j to its ancestor p;
p_j	:	the parent of node v_j in the routing tree.

The utilities of holding and transmitting a packet pkt at a node v_j depend on the following parameters related to traffic pattern:

- With respect to v_j itself and its children:

t_l	:	expected time to receive another packet pkt' from a child or locally from an upper layer;
s_l	:	expected payload size of pkt'.

- With respect to the parent of v_j:

t_p	:	expected time till the parent transmits another packet pkt'' that does not contain information elements generated or forwarded by v_j itself;
s_p	:	expected payload size of pkt''.

The utilities of holding and transmitting a packet pkt also depend on the following constraints posed by timeliness requirement for data delivery as well as limited packet size:

- Grace period t'_f for delivering pkt: the maximum allowable latency in delivering pkt minus the expected time taken to transport pkt from v_j to the sink without being held at any intermediate node along the route.
  If $t'_f \leq 0$, pkt should be transmitted immediately to minimize the extra delivery latency.
- Spare packet space s'_f of pkt: the maximum allowable payload length per packet minus the current payload length of pkt.
  Parameter s'_f and the size of the packets coming next from an upper layer at v_j or from v_j's children determine how much pkt will be packed and thus the potential utility of locally holding pkt.

Then, the utilities of holding and transmitting a packet are calculated as follows.

Utility of holding a packet. When a node v_j holds a packet pkt, pkt can be packed with incoming packets from v_j's children or from an upper layer at v_j. Therefore, the utility of holding pkt at v_j is the expected reduction in the amortized cost of transmitting pkt after packing pkt. The utility depends on (a) the expected number of packets that v_j will receive within t'_f time (either from a child or locally from an upper layer), and (b) the expected payload size s_l of these packets. Given that the expected inter-packet interval is t_l, the expected number of packets to be received at v_j within t'_f time is $\frac{t'_f}{t_l}$. Thus, the expected overall size $\mathcal{S}'_l$ of the payload to be received within t'_f time is $\frac{t'_f}{t_l}s_l$. Given the spare space s'_f in the packet pkt, the expected size $\mathcal{S}_l$ of the payload that can be packed into pkt is $\min\{\mathcal{S}'_l, s'_f\} = \min\{\frac{t'_f}{t_l}s_l, s'_f\}$.

Therefore, the expected amortized cost AC_l of transporting the packet to the sink R after the anticipated packing is

$$AC_l = \frac{1}{L - s'_f + \mathcal{S}_l} ETX_{jR}(L - s'_f + \mathcal{S}_l)$$

where $(L - s'_f)$ is the payload length of pkt before packing.

Since the amortized cost AC'_l of transporting pkt without the anticipated packing is

$$AC'_l = \frac{1}{L - s'_f} ETX_{jR}(L - s'_f)$$

the utility U_l of holding pkt is

$$U_l = AC'_l - AC_l \tag{3}$$

Utility of immediately transmitting a packet. If node v_j transmits the packet pkt immediately to its parent p_j, the utility comes from the expected reduction in the amortized cost of packet transmissions at p_j as a result of receiving the payload carried by pkt. When v_j transmits pkt to p_j, the grace period of pkt at p_j is still t'_f, thus the expected number of packets that do not contain information elements from v_j and can be packed with pkt at p_j is $\frac{t'_f}{t_p}$, and we use P_{pkt} to denote this set of packets. Given the limited payload that pkt carries, it may happen that not every packet in P_{pkt} gets packed (to full) via the payload from pkt. Accordingly, the utility U_p of immediately transmitting pkt is calculated as follows:

- If *every packet in P_{pkt} gets packed to full* with payload from pkt, i.e., $\frac{t'_f}{t_p}(L - s_p) \leq L - s'_f$:
  Then, the overall utility U'_p is

$$\begin{aligned} U'_p &= \frac{\frac{t'_f}{t_p} ETX_{p_j R}(s_p)}{\frac{t'_f}{t_p} s_p} - \frac{\frac{t'_f}{t_p} ETX_{p_j R}(L)}{\frac{t'_f}{t_p} L} \\ &= \frac{ETX_{p_j R}(s_p)}{s_p} - \frac{ETX_{p_j R}(L)}{L} \end{aligned} \tag{4}$$

- If *not every packet in P_{pkt} gets packed to full* with payload from pkt, i.e., $\frac{t'_f}{t_p}(L - s_p) > L - s'_f$:
  In this case, $\lfloor \frac{L - s'_f}{L - s_p} \rfloor$ number of packets are packed to full; if $\text{mod}(L - s'_f, L - s_p) > 0$, there is also a packet that gets partially packed with $\text{mod}(L - s'_f, L - s_p)$ length of payload from pkt. Thus the total number of packets that benefit from the packet transmission is $\lceil \frac{L - s'_f}{L - s_p} \rceil$. Denoting $\text{mod}(L - s'_f, L - s_p)$ by l_{mod} and letting I_{mod} be 1 if $l_{mod} > 0$ and 0 otherwise, then the overall utility U''_p is

$$U''_p = \frac{\lceil \frac{L - s'_f}{L - s_p} \rceil ETX_{p_j R}(s_p)}{\lceil \frac{L - s'_f}{L - s_p} \rceil s_p} - \frac{\lfloor \frac{L - s'_f}{L - s_p} \rfloor ETX_{p_j R}(L) + I_{mod} ETX_{p_j R}(s_p + l_{mod})}{\lceil \frac{L - s'_f}{L - s_p} \rceil s_p + L - s'_f} \tag{5}$$

Therefore, the utility U_p of immediately transmitting pkt to p_j is

$$U_p = \begin{cases} U'_p & \text{if } \frac{t'_f}{t_p}(L - s_p) \leq L - s'_f \\ U''_p & \text{otherwise} \end{cases} \tag{6}$$

where U'_p and U''_p are defined in Equations (4) and (5) respectively.

B. Scheduling rule

Given a packet to be scheduled for transmission, if the probability that the packet is immediately transmitted is P_t ($0 \leq P_t \leq 1$), then the expected utility $U_t(P_t)$ is

$$U_t(P_t) = P_t \times U_p + (1 - P_t) U_l = U_l + P_t(U_p - U_l) \tag{7}$$

where U_p and U_l are the utilities of immediately transmitting and locally holding the packet respectively. To maximize U_t, P_t should be set according to the following rule:

$$P_t = \begin{cases} 1 & \text{if } U_p > U_l \\ 0 & \text{otherwise} \end{cases}$$

That is, the packet should be immediately transmitted if the utility of immediate transmission is greater than that of locally holding the packet. For convenience, we call

this local, distributed decision rule *tPack* (for *time-sensitive packing*).

Competitive analysis. To understand the performance of tPack as compared with an optimal online algorithm, we analyze the competitive ratio of tPack. Since it is difficult to analyze the competitive ratio of non-oblivious online algorithms for arbitrary network and traffic pattern in the joint optimization [4] and tPack is a non-oblivious algorithm, we only study the competitive ratio of tPack for complete binary trees where all the leaf nodes generate information elements according to a common data generation process, and we do not consider the impact of packet length on link ETX. We denote these special cases of problem $\mathbb{P}$ as problem $\mathbb{P}'$. The theoretical analysis here is to get an intuitive understanding of the performance of tPack; we experimentally analyze the behaviors of tPack with different networks, traffic patterns, and application requirements through testbed based measurement in Section V. We relegate the study on the competitive ratio of tPack as well as the lower bound on the competitive ratio of non-oblivious online algorithms for the general problem $\mathbb{P}$ as a part of our future work. (Note that the best results so far on the lower bound of the competitive ratio of joint INP- and latency- optimization also only considered the cases where only leaf nodes generate information elements, and these results are for oblivious algorithms and for cases where no aggregation constraint is considered [4].)

Then, we have

Theorem 5: For problem $\mathbb{P}'$, tPack is $\min\{K, \max_{v_j \in V_{>1}} \frac{ETX_{v_j R}}{ETX_{p_j R}}\}$-competitive, where K is the maximum number of information elements that can be packed into a single packet, $V_{>1}$ is the set of nodes that are at least two hops away from the sink R.

Proof: Interested readers can find the proof in [11]. ■

From Theorem 5, we see that tPack is 2-competitive if every link in the network is of equal ETX value.

Implementation. Interested readers can find the discussion on how to implement tPack in TinyOS in [11].

V. Performance evaluation

To characterize the impact of packet packing and its joint optimization with data delivery timeliness, we experimentally evaluate the performance of tPack.

A. Methodology

Testbed. We use the *NetEye* wireless sensor network testbed at Wayne State University [16]. NetEye is deployed in an indoor office as shown in Figure 3. We use a 10×13 grid of TelosB motes in NetEye, where every two closest neighboring motes are separated by 2 feet. Out of the 130 motes in NetEye, we randomly select 120 motes (with each mote being selected with equal probability) to form a random network for our experimentation. Each of these TelosB motes is equipped with a 3dB signal attenuator and a 2.45GHz monopole antenna.

In our measurement study, we set the radio transmission power to be -25dBm (i.e., power level 3 in TinyOS) such that multihop networks can be created. We also use channel 26 of the CC2420 radio to avoid external interference from sources such as the campus WLANs. We use the TinyOS collection-tree-protocol (CTP) [17] as the routing protocol to form the routing structure, and we use the Iowa's Timesync protocol [18] for network wide time synchronization.

Figure 3. *NetEye* wireless sensor network testbed

Protocols studied. To understand the impact of packet packing and its joint optimization with data delivery timeliness, we comparatively study the following protocols:

- *noPack*: information elements are delivered without being packed in the network.
- *simplePack*: information elements are packed if they happen to be buffered in the same queue, but there is not packing-oriented scheduling.
- *tPack*: the packing- and timeliness-oriented scheduling algorithm that maximizes the local utility at each node, as we discussed in Section IV. (We have also evaluated another version of tPack, denoted by *tPack-2hop*, where the forwarding utility U_p considers both the parent node and the parent's parent; we find that tPack-2hop does not bring significant improvement over tPack while introducing higher overhead and complexity, thus our discussion here only focuses on tPack.)

We have implemented, in TinyOS [19], a system library which includes all the above protocols. The implementation takes 40 bytes of RAM (plus the memory required for regular packet buffers) and 4,814 bytes of ROM.

Performance metrics. For each protocol we study, we evaluate their behaviors based on the following metrics:

- *Packing ratio*: number of information elements carried in a packet;
- *Delivery reliability*: percentage of information elements correctly received by the sink;
- *Delivery cost*: number of transmissions required for delivering an information element from its source to the sink;
- *Latency jitter*: variability of the time taken to deliver information elements from the same source node, measured by the coefficient-of-variation (COV) [20] of information delivery latency.

Traffic pattern. To experiment with different sensornet scenarios, we use both periodic data collection traffic and event detection traffic trace as follows:

- $D6$: each source node periodically generates 50 infor-

mation elements with an inter-element interval, denoted by Δ_r, uniformly distributed between 500ms and 6s; this is to represent high traffic load scenarios.

- E_{lites}: an event traffic where a source node generates one packet based on the Lites [21] sensornet event traffic trace.

To understand the impact of the timeliness requirement of data delivery, we experiment with different latency requirements. For periodic traffic, we consider maximum allowable latency in delivering information elements that is 1, 5, 9, or 14 times the average element generation period, and we denote them by $L1$, $L5$, $L9$, and $L14$ respectively; for event traffic, we consider maximum allowable latency that is 3s, 12s, 36s, or 64s, and we denote them by $L3'$, $L12'$, $L36'$, and $L64'$ respectively. Out of the 120 motes selected for experimentation, we let the mote closest to a corner of NetEye be the sink node, and the other mote serves as a traffic source if its node ID is even. For convenience, we regard a specific combination of source traffic model and latency requirement a *traffic pattern*. Thus we have 8 traffic patterns in total. To gain statistical insight, we repeat each periodic traffic pattern 5 times and each event traffic pattern 10 times in our experiments. In the experiments, each information element is 16-byte long, and a packet can aggregate up to 7 information elements (i.e., $K = 7$).

B. Measurement results

Figures 4, 5, and 6 show the medians as well as their

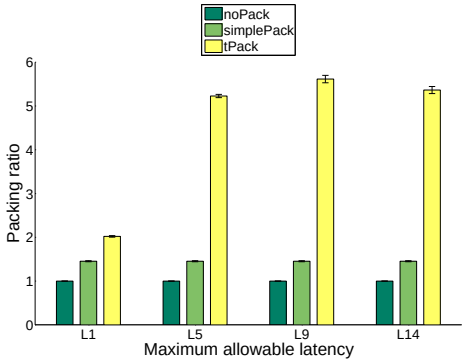

Figure 4. Packing ratio: $D6$

95% confidence level confidence intervals for the packing ratio, delivery reliability, and delivery cost when the source traffic model is $D6$. (Similar phenomena are observed for E_{lites}, and we relegate the details to [11] due to the limitation of space.) We see that the packing ratio in tPack is significantly higher than that in noPack and simplePack. The increased packing ratio reduces channel contention and thus reduces the probability of packet transmission collision, which improves data delivery reliability. The reduced probability of transmission collision and the increased number of information elements carried per packet in tPack in turn reduces delivery cost, since there are fewer number of packet

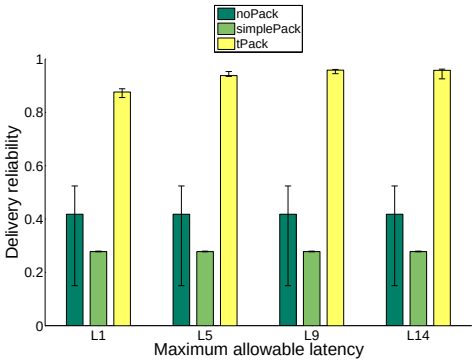

Figure 5. Delivery reliability: $D6$

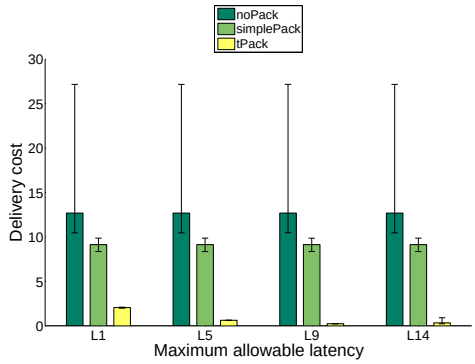

Figure 6. Delivery cost: $D6$

retransmissions as well as fewer number of packets generated. Note that the low delivery reliability in simplePack is due to intense channel contention.

Figure 5 also shows that tPack improves data delivery reliability even when the allowable latency in data delivery is small (e..g, in the case of $L1$) where the inherent probability for packets to be packed tends to be small. Therefore, tPack can be used for real-time applications where high data delivery reliability is desirable. Figure 4 shows that the packing ratio in tPack is very high and close to 6 except for the case of $L1$ where the packing probability is significantly reduced by the limited probability for a node to wait due to stringent timeliness requirement. Note that the upper bound on the packing ratio in our experiments is 7, thus tPack achieves a packing ratio very close to the optimal, which corroborates our analytical result in Theorem 5.

These figures show, surprisingly, that simplePack can perform worse than noPack despite packet packing in simplePack. This is because packet packing in simplePack can be too limited to significantly reduce channel contention such that the increased packet length as a result of the packing actually reduces packet delivery reliability; this is further exacerbated by the fact that losing a packed packet means losing more information elements than that of an unpacked packet containing only one information element. Therefore, it is important to schedule packet transmissions

to improve the degree of packet packing so that the drawbacks of packing can be overshadowed by the benefits. On the other hand, the difference between the performance of simplePack and noPack is not statistically significant at the 95% confidence level. Moreover, packet packing in simplePack still reduces the delivery cost of noPack as can be seen from Figure 6, which is desirable in low-power sensornets. Due to the reduced contention in simplePack and the fact that co-channel contention increases uncertainty in data delivery, network performance has lower variability and is more predictable (e.g., narrower confidence intervals for the medians of performance metrics) in simplePack than in noPack too.

Similarly, performance variability is low in tPack due to the reduced channel contention which is in turn a result of the improved packet packing. For instance, Figure 7 shows the latency jitter in different protocols, and we see

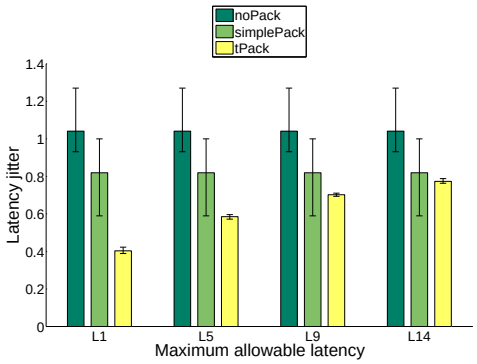

Figure 7. Latency jitter: $D6$

that the jitter is the lowest in tPack. In tPack, moreover, the latency jitter decreases as allowable latency decreases. These properties are desirable in CPS sensornets where real-time sensing and control require predictable data delivery performance (e.g., in terms of low latency jitter), especially in the presence of potentially unpredictable, transient perturbations.

VI. Related work

In-network processing (INP) has been well studied in sensornets, and many INP methods have been proposed for query processing (e.g., TinyDB [1]) and general data collection (e.g., DFuse [2]). When controlling spatial and temporal data flow to enhance INP, however, these methods did not consider application requirements on the timeliness of data delivery. As a first step toward understanding the interaction between INP and application QoS requirements, our study has shown the benefits as well as the challenges of jointly optimizing INP and QoS from the perspective of packet packing. As sensornets are increasingly being deployed for mission-critical tasks, it becomes important to address the impact of QoS requirements on general INP methods other than packet packing, which opens interesting avenues for further research.

As a special INP method, packet packing has also been studied for sensornets as well as general wireless and wired networks, where mechanisms have been proposed to adjust the degree of packet packing according to network congestion level [5], [22], to address MAC/link issues related to packet packing [23], [6], [24], to enable IP level packet packing [25], and to pack periodic data frames in automotive applications [26]. These works have focused on issues in local, one-hop networks without considering requirements on maximum end-to-end packet delivery latency in multi-hop networks. With the exception of [26], these works did not focus on scheduling packet transmissions to improve the degree of packet packing, and they have not studied the impact of finite packet size either. Saket et al. [26] studied packet packing in single-hop controller-area-networks (CAN) with finite packet size. Our work addresses the open questions on the complexity and protocol design issues for jointly optimizing packet packing and data delivery timeliness in multi-hop wireless sensornets.

Most closely related to our work is [3] where the authors studied the issue of optimizing INP under the constraint of maximum end-to-end data delivery latency. But the study did not consider aggregation constraints and instead assumed *total aggregation* where any arbitrary number of information elements can be aggregated into one single packet. The study did not evaluate the impact of joint optimization on data delivery performance either. Our work focuses on settings where packet size is finite, and we show that aggregation constraints (in particular, maximum packet size and re-aggregation tolerance) significantly affect the problem complexity and protocol design. Using a high-fidelity sensornet testbed, we also systematically examine the impact of joint optimization on packet delivery performance in multi-hop wireless networks.

Solis et al. [27] also considered the impact that the timing of packet transmission has on data aggregation, and the problem of minimizing the sum of data transmission cost and delay cost has been considered in [4] and [28]. These studies also assumed total aggregation, and they did not consider hard real-time requirements on maximum end-to-end data delivery latency. Ye et al. [29] considered the local optimal stopping rule for data sampling and transmission in distributed data aggregation. It did not consider hard real-time requirement either, and it did not study network-wide coordination and the limit of data aggregation. Yu et al. [30] studied the latency-energy tradeoff in sensornet data gathering by adapting radio transmission rate; it did not study the issue of scheduling data transmission to improve the degree of data aggregation.

VII. Concluding remarks

Through both theoretical and experimental analysis, we examine the complexity and impact of jointly optimizing packet packing and the timeliness of data delivery. We find that aggregation constraints (in particular, maximum packet size and re-aggregation tolerance) affect the problem complexity more than network and traffic properties do,

which suggest the importance of considering aggregation constraints in the joint optimization. We identify conditions for the joint optimization to be strong NP-hard and conditions for it to be solvable in polynomial time. For cases when it is polynomial-time solvable, we solve the problem by transforming it to the maximum weighted matching problem in interval graphs; for cases when it is strong NP-hard, we prove that there is no polynomial-time approximation scheme (PTAS) for the problem. We also develop a local, distributed online protocol tPack for maximizing the local utility of each node, and we prove the competitiveness of the protocol with respect to optimal solutions. Our testbed based measurement study also corroborates the importance of QoS- and aggregation-constraint aware optimization of packet packing.

While this paper has extensively studied the complexity, algorithm design, and impact of jointly optimizing packet packing and data delivery timeliness, there are still a rich set of open problems. Even though we have analyzed the competitiveness of tPack for non-trivial scenarios and this has given us insight into the behavior of tPack, it remains an open question on how to characterize in a closed form the competitiveness of tPack and non-oblivious online algorithms in broader contexts. The analytical and algorithmic design mechanisms developed for packet packing may well be extensible to address other in-network processing methods such as data fusion, and a detailed study of this will help us better understand the structure of the joint optimization problem and will be interesting future work to pursue. We have focused on the scheduling aspect of the joint optimization, and we are able to use mathematical tools such as interval graphs to model the problem; on the other hand, how to mathematically model and analyze the impact of the joint optimization on spatial data flow is still an open question and is beyond the scope of most existing network flow theory, thus it will be interesting to explore new approaches to modeling and solving the joint optimization problem.

REFERENCES

[1] S. Madden, M. Franklin, and J. Hellerstein, "TinyDB: An acquisitional query processing system for sensor systems," in *ACM Transactions on Database Systems*, 2004.

[2] R. Kumar, M. Wolenetz, B. Agarwalla, J. Shin, P. Hutto, A. Paul, and U. Ramachandran, "DFuse: A framework for distributed data fusion," in *ACM SenSys*, 2003.

[3] L. Becchetti, P. Korteweg, A. Marchetti-Spaccamela, M. Skutella, L. Stougie, and A. Vitaletti, "Latency constrained aggregation in sensor networks," in *European Symposium on Algorithms (ESA)*, 2006.

[4] Y. A. Oswald, S. Schmid, and R. Wattenhofer, "Tight bounds for delay-sensitive aggregation," in *ACM PODC*, 2008.

[5] T. He, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "AIDA: Adaptive application independent data aggregation in wireless sensor networks," *ACM Transaction on Embedded Computing System*, vol. May, 2004.

[6] K. Lu, D. Wu, Y. Qian, Y. Fang, and R. C. Qiu, "Performance of an aggregation-based MAC protocol for high-data-rate ultrawideband ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 56, no. 1, pp. 312–321, 2007.

[7] A. Arora et al., "A line in the sand: A wireless sensor network for target detection, classification, and tracking," *Computer Networks (Elsevier)*, vol. 46, no. 5, 2004.

[8] "IETF 6LowPAN working group," http://www.ietf.org/html.charters/6lowpan-charter.html.

[9] "IETF ROLL working group," http://www.ietf.org/html.charters/roll-charter.html.

[10] G. Finke, V. Jost, M. Queyranne, and A. Sebo, "Batch processing with interval graph compatibilities between tasks," *Discrete Applied Mathematics (Elsevier)*, vol. 156, 2008.

[11] Q. Xiang, J. Xu, X. Liu, H. Zhang, and L. J. Rittle, "When in-network processing meets time: Complexity and effects of joint optimization in wireless sensor networks," Wayne State University (http://www.cs.wayne.edu/∼hzhang/group/TR/DNC-TR-09-01.pdf), Tech. Rep., 2009.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of* NP*-Completeness*. Freeman and Company, 1979.

[13] D. S. Hochbaum, *Approximation Algorithms for* NP*-hard Problems*. PWS Publishing Company, 1997.

[14] P. Wang, J. Zheng, and C. Li, "Data aggregation using distributed lossy source coding in wireless sensor networks," in *IEEE GLOBECOM*, 2007.

[15] H. Gabow, "An efficient implementation of edmonds' algorithm for maximum matchings on graphs," *Journal of ACM*, vol. 23, pp. 221–234, 1975.

[16] "NetEye testbed," http://www.cs.wayne.edu/∼hzhang/group/systems-and-downloads.html#NetEye.

[17] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," Stanford University, Tech. Rep. SING-09-01, 2009.

[18] "Iowa's timesync component," http://tinyos.cvs.sourceforge.net/viewvc/tinyos/tinyos-2.x-contrib/iowa/T2.tsync/.

[19] "TinyOS," http://www.tinyos.net/.

[20] R. Jain, *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, Inc., 1991.

[21] "An event traffic trace for sensor networks," http://www.cs.wayne.edu/∼hzhang/group/publications/Lites-trace.txt.

[22] A. Jain, M. Gruteser, M. Neufeld, and D. Grunwald, "Benefits of packet aggregation in ad-hoc wireless network," University of Colorado at Boulder, Tech. Rep. CU-CS-960-03, 2003.

[23] T. Li, Q. Ni, D. Malone, D. Leith, Y. Xiao, and T. Turletti, "Aggregation with fragment retransmission for very high-speed WLANs," *IEEE/ACM Transactions on Networking*, vol. 17, no. 2, pp. 591–604, 2009.

[24] M. Li, H. Zhu, Y. Xiao, I. Chlamtac, and B. Prabhakaran, "Adaptive frame concatenation mechanisms for qos in multi-rate wireless ad hoc networks," in *IEEE INFOCOM*, 2008.

[25] D. Kliazovich and F. Granelli, "Packet concatenation at the ip level for performance enhancement in wireless local area networks," *Wireless Networks*, vol. 14, pp. 519–529, 2008.

[26] R. Saket and N. Navet, "Frame packing algorithms for automotive applications," *Journal of Embedded Computing*, vol. 2, pp. 93–102, 2006.

[27] I. Solis and K. Obraczka, "The impact of timing in data aggregation for sensor networks," in *IEEE ICC*, 2004.

[28] A. R. Karlin, C. Kenyon, and D. Randall, "Dynamic TCP acknowledgement and other stories about $\frac{e}{e-1}$," in *ACM STOC*, 2001.

[29] Z. Ye, A. A. Abouzeid, and J. Ai, "Optimal policies for distributed data aggregation in wireless sensor networks," in *IEEE INFOCOM*, 2007.

[30] Y. Yu, V. Prasanna, and B. Krishnamachari, "Energy minimization for real-time data gathering in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 5, no. 11, 2006.