Maximizing Information in Unreliable Sensor Networks under Deadline and Energy Constraints

Srikanth Hariharan*, Zizhan Zheng and Ness B. Shroff

***Abstract*—We study the problem of maximizing the information in a wireless sensor network with unreliable links. We consider a sensor network with a tree topology, where the root corresponds to the sink, and the rest of the network detects an event and transmits data to the sink. We formulate a combinatorial optimization problem that maximizes the *information* that reaches the sink under deadline, energy, and interference constraints. This framework allows using a variety of error recovery schemes to tackle link unreliability. We show that this optimization problem is NP-hard in the strong sense when the input is the maximum node degree of the tree. We then propose a dynamic programming framework for solving the problem exactly, which involves solving a special case of the Job Interval Selection Problem (JISP) at each node. Our solution has a polynomial time complexity when the maximum node degree is $O(\log N)$ in a tree with N nodes. For trees with higher node degrees, we further develop a sub-optimal solution, which has low complexity and allows distributed implementation. We investigate tree structures for which this solution is optimal to the original problem. The efficiency of the sub-optimal solution is further demonstrated through numerical results on general trees.**

I. Introduction

A wireless sensor network is a wireless network consisting of a number of sensors that sense a desired aspect of the region in which they are deployed. These networks are used in a number of military and civilian applications, such as target tracking and environment monitoring. Sensor measurements are prone to errors due to environmental factors and resource constraints. Therefore, sinks cannot rely on the data sensed by a single sensor. In many applications, the sinks only desire a certain function of the data sensed by different sensor nodes (e.g., average temperature, maximum pressure, detect a signal, etc.). When sinks require certain classes of functions of the sensed data, performing *in-network computation* (intermediate nodes in the network aggregate data from all their predecessors, and only transmit the aggregated data) greatly reduces the communication overhead [2].

A tree structure is commonly used for data aggregation in wireless sensor networks [3], [4]. In this paper, we consider a tree topology with the sink as the root of the tree. An event is observed by a subset of nodes in the tree called the source nodes. All source nodes transmit their data about the event to the sink. Our goal is to maximize the *information* obtained by the sink. The information obtained by the sink is a representation of the quality of the data that reaches the sink. For example, it could be the sum of the inverses of the error variances of the data from various sources that reaches the sink [5]. It could also represent other relevant metrics such as the Log-Likelihood Ratio if detection is being performed by the network, distortion, etc.

Much of the existing work in data aggregation does not take channel errors, and interference into account. However, wireless channels are inherently prone to errors due to fading and environmental factors. Also, interference is a critical component of the wireless environment. We consider a one-hop interference model where two nodes that are one hop away from each other cannot transmit simultaneously. We also consider unreliable links where the errors across different links are independent of each other, and allow for the usage of various error-recovery schemes including retransmissions, coding, etc.

Delay is also an important parameter in a wireless sensor network. While most works focus only on energy, minimizing the delay can help save a huge amount of energy. For instance, suppose that a sensor network is tracking a target. In order to ensure good tracking quality, the sink must obtain previous measurements in a timely manner so that the best subset of sensors for the next measurement is chosen. If the sink does not get the measurements in a timely manner, the target might have moved too far resulting in a poor measurement quality during future measurements. On the other hand, if the sink decides to ensure good tracking quality by activating a large number of sensors at all times, a large amount of energy could be wasted. Therefore, it is critical that the sink obtains sensor measurements in a delay efficient manner.

With this model, we provide an optimization framework for maximizing the information received at the sink under a deadline constraint at the sink, and per-sensor energy constraints. The output of this framework is a schedule of time slots for each node within the deadline, and the amount of energy to be expended by each node on transmissions and receptions.

The main contributions of this work are summarized as

This work was supported in part by ARO MURI Awards W911NF-07-10376 (SA08-03) and W911NF-08-1-0238, and NSF Awards 0626703-CNS, 0635202-CCF, and 0721236-CNS.

S. Hariharan is with the Department of Electrical and Computer Engineering, The Ohio State University, 2015 Neil Ave., Columbus, OH 43210, USA srikanth.hariharan@gmail.com

Z. Zheng is with the Department of Electrical and Computer Engineering, The Ohio State University, 2015 Neil Ave., Columbus, OH 43210, USA zhengz@ece.osu.edu

N. B. Shroff is with the Department of Electrical and Computer Engineering and the Department of Computer Science and Engineering, The Ohio State University, 2015 Neil Ave., Columbus, OH 43210, USA shroff@ece.osu.edu

* Corresponding author.

A preliminary version of this paper by S. Hariharan and N. B. Shroff titled "Deadline Constrained Scheduling for Data Aggregation in Unreliable Sensor Networks" appeared in the proceedings of the 9^{th} Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WIOPT), 2011 [1].

follows:

- We develop an optimization based framework to maximize the information that is received at the sink from all the source nodes in the data gathering tree. This optimization framework *explicitly accounts for unreliable links, deadlines, per-sensor energy constraints, and interference*.
- We show that this optimization problem is NP-hard in the strong sense by a reduction from the 3-partition problem [6], which implies that the problem does not allow a pseudo-polynomial time algorithm or a fully polynomial time approximation scheme (FPTAS) [6].
- We develop a dynamic programming framework to solve the optimization problem, which involves solving a special case of the single-machine Job Interval Selection Problem (JISP) at each node. For this subproblem, we propose an optimal solution using dynamic programming, which has an exponential time complexity of $O(2^k)$, where k denotes the maximum node degree in the data gathering tree. This leads to a polynomial time solution to our problem when $k = O(\log N)$, where N denotes the total number of nodes, a condition that often holds in practice.
- For a dense sensor network with high node degrees, we further develop a sub-optimal solution for solving the interval scheduling subproblem, which has low complexity and allows distributed implementation. This solution is optimal to the original problem for certain tree structures.
- We evaluate the performance of the sub-optimal algorithm for general tree structures through numerical evaluations, and show that it not only performs close to the optimal solution but also outperforms an existing JISP approximation algorithm.

The rest of this paper is organized as follows. In Section II, we overview related work. In Section III, we describe our system model and assumptions. In Section IV, we formulate our problem. In Sections V, VI, and VII, we study the problem without energy constraints. In Section V, we study the structural properties of the deadline constrained problem, and prove that it is NP-hard in the strong sense. We then discuss an optimal solution based on dynamic programming in Section VI. In Section VII, we propose a sub-optimal solution to the deadline constrained problem, and show its optimality for certain tree structures. In Section VIII, we then extend our solutions to the general problem formulated in Section IV (including energy constraints). In Section IX, we provide numerical results comparing the performance of our sub-optimal algorithm with that of the optimal solution and a baseline algorithm for general tree structures. Finally, in Section X, we conclude the paper.

II. Related Work

The problem of evacuating all the packets in a multi-hop wireless network in minimum time is quite related to what we are studying. Different works [7], [8] have studied this problem for error-free channels. They propose polynomial-time algorithms for tree structures for the one-hop interference model. These works do not consider in-network computation.

Other existing work in this area can be broadly characterized into two classes - constructing efficient data aggregation trees [3], [9] and approaches to study the trade-offs between energy, delay and the quality of aggregated data [10], [11], [12], [13], [1], [14]. Constructing an optimal data aggregation tree is NP-hard for a number of cases such as minimizing the number of transmissions [9], maximizing the lifetime [3], etc. Most of these works considering the construction of data gathering trees do not take interference and unreliable channels into account. Our focus in this work is to design a communication-efficient protocol in any given data gathering tree.

Several existing works use optimization approaches to study energy-delay trade-offs and energy-quality trade-offs under data aggregation. Boulis et. al., [10] study trade-offs between energy and data accuracy in data aggregation trees. In [11], Yu et. al., study trade-offs between energy and latency in data aggregation trees, assuming a time-slotted synchronized system. In [12], Ye et. al., study the fundamental energy-delay trade-off for distributed data aggregation in wireless sensor networks. Their goal is to maximize a certain parameter called the discounted reward at each node, where the reward is due to data aggregation and the discount is due to the time for which the node waits in order to aggregate data from its predecessors. The common drawback in all of these works is that they do not take link errors and interference constraints into account while framing their optimization problem. We have previously studied the problem of maximizing information in data gathering trees under deadline and one-hop interference constraints, *for error-free links* [13]. For unit capacity links, a distributed optimal solution was developed that involved solving a localized Maximum Weighted Matching (MWM) problem at each hop. Since the matching only involved neighboring nodes, the algorithm had a low computational complexity. However, the inclusion of link errors, as done in this work, substantially increases the difficulty of the problem. We have also studied a problem of maximizing information under energy constraints in [14].

In a preliminary version of this work [1], we have studied maximizing the information for unreliable sensor networks considering deadline constraints only. Here, we also consider energy constraints, show that the problem is NP-hard in the strong sense, and develop a more efficient optimal solution ($O(2^k)$) than the exhaustive search based optimal solution ($O(k!)$) in [1].

III. System Model and Assumptions

We model the system as a graph $G(V \cup \{S\}, E)$ where V is the set of *active* nodes, S is the sink, and E is the set of links. The graph G is a tree rooted at the sink. An *active* node is one that is either a source node or has at least one source node in its sub-tree. Sensors take measurements for *events* (such as tracking a target, measuring the temperature, etc.), and send an aggregated form of the data to the sink within a deadline. A node may or may not be a source for a particular event.

We consider a time-slotted and synchronized system. We assume that the number of time slots to transmit a packet, the number of transmissions (including retransmissions, number

of coded bits, etc.) are all integers. Link capacities need not be identical across links. For simplicity of exposition, we assume that each source has data ready at time zero. It is straightforward to extend this notion to incorporate arbitrary known time slots at which data is ready at nodes [13]. We assume that each source has a single data packet for an event, and the next event occurs only after the deadline for the current event has expired. We also consider a primary (or one-hop) interference model where no two links that share a node can be active at the same time. This model has been used to characterize the interference for Bluetooth or FH-CDMA based systems. Extending the results to a more general class of interference models is an open problem for future work.

We assume "perfect" aggregation, i.e., intermediate nodes can gather data from predecessors, and aggregate them into a single packet [2], [4], [9], [13]. For reliable links and energy constraints, we have also extended this framework for "imperfect" aggregation [15]. Further, in [15], we have also extended our solution to account for optimal selection of source nodes to maximize information.

We now define the following notations (Table I).

V	Set of nodes
S	Sink
E	Set of links
V_L	Set of leaf nodes
V_S	Set of source nodes
$P(i)$	Parent of node i
$C(i)$	Set of children of node i
w_i	Information provided by source node i
w_i^j	Information received at node i from node j
t_i	Number of time slots allocated to node i for transmissions
W_i	Number of time slots that node i waits to receive packets
$\mathcal{T}_i$	Maximum number of time slots that can be allocated to node i for transmissions
D	Deadline at sink
e_i	Energy allocated to node i for transmissions
E_i	Energy constraint at node i for transmissions and receptions
E_T	Energy expended on one transmission
E_R	Energy expended on one reception
$\mathcal{E}_i$	Maximum energy that can be allocated to node i for transmissions
$f_i(t_i, e_i)$	Link reliability function of the link from i to its parent

TABLE I
NOTATIONS

We refer to the actual sensor measurements as *data*. For each source node i, we define the *information*, w_i, provided by i as follows. Let $\vec{x_i}$ (of dimension n_i) represent the raw measurements of sensor i. We define w_i as $I(\vec{x_i})$, where $I : \mathbb{R}^{n_i} \rightarrow \mathbb{R}$. The function $I(\cdot)$ represents the quality of the data. For example, data could be temperature, location, etc., while information could represent error variance, distortion, Log-Likelihood Ratio, etc. By aggregated data, we mean the in-network computed data (or the information provided by the in-network computed data).

Each sensor has a per-sensor energy constraint. The amount of energy spent by a sensor for any event is proportional to the amount of energy spent on transmissions and receptions.

Wireless links are unreliable, and we assume that packets losses are independent across links. Since links are unreliable, we can employ a number of known error-recovery techniques such as retransmissions, coding, etc. We model the *link reliability* of a link from node i (to its parent) as a function $f_i(t, e)$, where t is the number of time slots allocated for transmissions for node i, and e is the energy spent on transmissions over the link. $f_i(t, e)$ denotes the probability that information is successfully transmitted from node i if t transmissions slots, and e units of energy for transmission are allowed. Note that $f_i(t, e)$ can be any arbitrary function (that is of practical importance), and hence can model any known error-recovery technique for that particular link. Similarly, it can account for different link capacities. For instance, if 2 time slots are required to transmit data from node i to its parent, and $t = 1$, $f_i(t, e)$ can be set to zero. Further, it also allows modeling various relationships between the energy required for transmissions, and the time slots allocated for transmission.

To illustrate the link reliability function, consider a simple case in which transmission during any given slot consumes a fixed amount of energy. Then, $f_i(t, e)$ can simply be a function of $\min(t, e)$. For example, if the link capacity is one, $E_T = 1$, and we use retransmissions, $f_i(t, e) = (1 - p^{\min(t,e)})$, where p is the packet error rate over the link from i to its parent. Here, $f_i(t, e)$ is the probability that the packet from i reaches its parent. Consider another example. Assume that the link capacity is one, $E_T = 1$, and that p represents the bit error rate. Suppose we use coding, and we require at least k bits to be successful for decoding at the receiver. Then, $f_i(t, e) = \sum_{j=k}^{\min(t,e)} \binom{\min(t,e)}{j} (1-p)^j p^{(\min(t,e)-j)}$ is the probability that at least k out of $\min(t, e)$ bits are successful.

We define the *information received at node B from a particular source node A*, denoted by w_B^A, as the product of the information provided by node A and the *product (or weighted product)* of the *link reliabilities* over all the links from node A to node B. To be precise, if $P_{A,B}$ represents the path from A to B, then the information received at B from A is $w_B^A = w_A \prod_{j \in P_{A,B}} f_j(t_j, e_j)$.

Remark: We note that the algorithms we develop in the later sections also work when the information received at node B from source node A is defined as the product of the information provided by node A and the *sum (or weighted sum)* of the *link reliabilities* over all the links from node A to node B, i.e., $w_B^A = w_A(\sum_{j \in P_{A,B}} f_j(t_j, e_j))$. The product of the link reliabilities is very meaningful when the link errors are independent. This, for instance, could represent the *expected* information that reaches the sink from any node in the tree. The sum of the link reliabilities is also of importance as it can be used to model fairness. For instance, we can model proportional fairness [16] among links using a logarithmic function. Note that the logarithm of the product of link reliabilities is the sum of the logarithms of the link reliabilities.

Finally, we model the *information received at node B* as a (weighted) sum of the information received at B from individual source nodes. Note that when B is the sink, we obtain the information received at the sink. This metric is of importance especially when sensor measurements are fused. For instance, the *sum* of the inverses of the error variances of individual sensors represents the overall error variance of the

fused measurements when measurements are independent [5].

A. Motivating application - sensor networks

Consider a sensor network shown in Figure 1. Sensors send their data over a tree network to a sink (red node). In order to illustrate our definitions of information, consider a part of this network (encircled in green) with three nodes and the sink S. Suppose that all the three sensors are source nodes measuring the location of a target. Let sensor i provide an error variance R_i for the target location, $i \in \{1, 2, 3\}$. Then, a metric for measuring the *information* provided by sensor i is $\frac{1}{R_i}$. If data from sensors 1 and 2 is combined (say), then one of the metrics for the overall variance, R_{12} of the combined data is given by $\frac{1}{R_{12}} = \frac{1}{R_1} + \frac{1}{R_2}$, which is lower than both R_1 and R_2 [5].

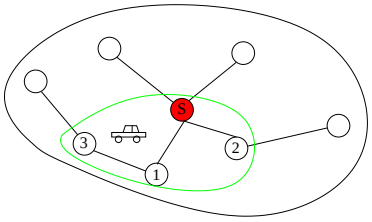


Fig. 1. Motivating Example

Suppose that retransmissions are used in order to account for unreliable links. If sensor i is allowed to make t_i transmissions and consume e_i units of energy on transmissions, the probability that sensor i's transmission is successful is given by $f_i(t_i, e_i)$. When data aggregation is used at intermediate nodes, for example, node 1 can combine data from node 3, and transmit both data simultaneously. Assuming that errors across links are independent, the probability that node 1's data reaches the sink is $f_1(t_1, e_1)$, the probability that node 2's data reaches the sink is $f_2(t_2, e_2)$, and the probability that node 3's data reaches the sink is $f_3(t_3, e_3)f_1(t_1, e_1)$. Now, the expected information received at the sink is $\sum_{i\in\{1,2,3\}} \frac{1}{R_i}$P($i$'s data reaches the sink) = $\frac{1}{R_1}f_1(t_1, e_1) + \frac{1}{R_2}f_2(t_2, e_2) + \frac{1}{R_3}f_3(t_3, e_3)f_1(t_1, e_1)$. One can clearly verify that without taking the communication model into account, we get the information $\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}$.

IV. Problem Formulation

We first describe our forwarding policy for data aggregation. **Forwarding Policy:** For an event, each node will wait for a certain time to aggregate data from its predecessors. Until that waiting time expires, a node, even if it is a source node, will not transmit its data to its parent. After the waiting time expires, the node will no longer accept transmissions from its children. The implication of this policy is that a node will never transmit data that it receives from two or more nodes separately. It will always aggregate the data that it receives and transmit the aggregated data.

This forwarding policy is commonly used in the literature for data gathering problems [2], [3], [9], [13], [14].

We now formulate our optimization problem. The notations below can be recalled from Table I.

Problem Π_{DE}:

$$\max_{\vec{t},\vec{W},\vec{e}} \quad \sum_{i\in V_S} w_S^i$$

$$\text{s.t.} \quad t_i \in \{0, 1, ..., \mathcal{T}_i\} \ \forall i \in V$$

$$e_i \in \{0, 1, ..., \mathcal{E}_i\} \ \forall i \in V$$

$$e_i E_T + \sum_{j\in C(i)} e_j E_R \leq E_i \ \forall i \in V \cup \{S\} \quad (1)$$

$$\text{For each } i \in V \cup \{S\}\backslash V_L, \text{ and for each } C \subseteq C(i),$$

$$\sum_{j:j\in C} t_j \leq W_i - \min_{j:j\in C} W_j \quad (2)$$

$$W_i \in \{0, 1, ..., D-1\} \ \forall i \in V \text{ and } W_S = D$$

We now explain the constraints in problem Π_{DE}. Constraint (1) represents the energy constraint at each node in the network. This consists of transmission and reception energies. We note that leaf nodes do not expend energy on receptions, and the sink does not expend energy on transmissions. The relation between one-hop interference and delay is represented by (2). Under the one-hop interference model, a parent node can only receive packets from one of its children nodes during a particular slot. However, when a child transmits to its parent, the other children (of the same parent) can receive data from their children (by the definition of the one-hop interference model). Constraint (2) says that for any subset of the children nodes, the total number of transmissions made by this subset of nodes is bounded above by the difference between the waiting time of the parent and the waiting time of the child that has the least waiting time in the chosen subset. This constraint is similar to the interference constraint for error-free channels. Examples illustrating this constraint can be found in [13].

While we are ultimately interested in solving problem Π_{DE}, it is interesting to first study the problem without energy constraints, i.e., with only deadline and interference constraints. Studying this problem provides insights into solving Π_{DE}. We call this problem without energy constraints Π_D. Without energy constraints, $\mathcal{E}_i$ units of energy can be expended on transmissions by each node i. For notational convenience, in Π_D, we refer to the link reliability function, $f_i(t_i, \mathcal{E}_i)$, as simply $f_i(t_i)$. We study the structural properties of Π_D and show that it is NP-hard. This automatically implies that the problem Π_{DE} is also NP-hard. We first solve Π_D, and use its solution to develop an algorithm that solves Π_{DE}.

V. Structural Properties and NP-hardness

In this section, we study the properties of the optimal solution of problem Π_D, and show that it is NP-hard. These properties will be used to derive an optimal algorithm in Section VI and a sub-optimal algorithm with lower complexity in Section VII.

Theorem V.1. *Consider any single hop in the data aggregation tree with parent node P having k children, C_1, C_2,..., C_k. For problem Π_D, let an optimal waiting time of P be W_P^* and let W_1^*,..., W_k^* be optimal waiting times of the k children, respectively. Let t_1^*, t_2^*,..., t_k^* be an optimal solution*

for the number of transmissions made by the k children, respectively. WLOG, assume that $W_1^* \leq W_2^* \leq ... \leq W_k^*$. For $j \in \{1, 2, ..., k\}$, define W'_j recursively as follows.

$$W'_1 = W_1^* \quad (3)$$

$$W'_j = \max(W_j^*, W'_{j-1} + t_{j-1}^*), \text{ if } j \in \{2, 3, ..., k\} \quad (4)$$

Then, for each $j \in \{1, 2, ..., k\}$, we have the following properties:

1) *W'_j is also an optimal waiting time for node C_j.*
2) *W'_j, $W'_j + 1$, ..., $W'_j + t_j^* - 1$ are optimal transmission slots for node C_j.*

Proof: We show 1) and 2) by induction on j.

Consider $j = 1$. By definition, $W'_1 = W_1^*$, and hence W'_1 is an optimal waiting time for node C_1. We prove 2) by contradiction. Suppose that W'_1, $W'_1 + 1$, ..., $W'_1 + t_1^* - 1$ are not optimal transmission slots for node C_1. Since C_1 cannot transmit before W'_1 because of the forwarding policy, and since C_1 needs to make t_1^* transmissions, C_1 must transmit in at least one time slot after $W'_1 + t_1^* - 1$. If there exists a time slot in $\{W'_1, W'_1 + 1, ..., W'_1 + t_1^* - 1\}$ during which none of the children of P transmit, then making C_1 transmit during this slot does not affect the optimality of the solution. Suppose that there exists a time slot in $\{W'_1, W'_1 + 1, ..., W'_1 + t_1^* - 1\}$ during which a child of P other than C_1 transmits. Once again, by making C_1 transmit during this slot, and scheduling the other child to transmit in the time slot after $W'_1 + t_1^* - 1$ (in which C_1 was originally transmitting), the optimality of the solution is not affected. This can be reasoned as follows.

- The new schedule is feasible.
- The value of the objective function does not decrease because of interchanging the schedules. This is because the expected number of packets aggregated by C_1 does not change after time slot W'_1, and the expected number of packets aggregated by any other node in that hop cannot decrease since it now has greater time to gather data from its predecessors.

This contradicts our assumption that W'_1, $W'_1 + 1$, ..., $W'_1 + t_1^* - 1$ are not optimal transmission slots for C_1.

Assume that 1) and 2) are true for node C_m.

Consider $j = m + 1$. If $\max(W_{m+1}^*, W'_m + t_m^*) = W_{m+1}^*$, then clearly W'_{m+1} is an optimal waiting time for node C_{m+1}. Suppose that $W_{m+1}^* < W'_m + t_m^*$. By 1) and 2) for node C_m, node C_{m+1} cannot start transmitting before $W'_m + t_m^*$. If node C_{m+1} waits until slot $W'_m + t_m^* (> W_{m+1}^*)$, it can potentially aggregate more packets, and still make t_{m+1}^* transmissions. Therefore, the value of the objective function cannot decrease if C_{m+1} waits until $W'_m + t_m^*$. Hence, 1) follows for node C_{m+1}.

The proof of 2) is virtually identical to that for the case $j = 1$.

Thus, by induction, 1) and 2) are true $\forall j \in \{1, 2, ..., k\}$. ■

Theorem V.1 shows that in order to find a collision-free optimal schedule, it is enough to know the optimal waiting time of each node, and the optimal number of time slots to be allocated for transmission over each link. Specifically, it shows that if the optimal waiting time of a node i is W_i, and the optimal number of time slots it is allocated is t_i, then the optimal collision-free schedule of i is the set of time slots $\{W_i, W_i + 1, ..., W_i + t_i - 1\}$.

We now show that π_D can be solved in a recursive manner.

Let $X[i, W]$ represent the information received at node i if node i waits for a time W. We define $X[i, W]$ as follows. For any leaf node l, for any $W \in \{0, 1, ..., D - 1\}$, we have $X[l, W] = w_l \lambda_l$. (Here, $\lambda_l = 1$, if l is a source, and zero, otherwise). Recall that w_l is the information provided by l. Consider any hop with parent node P having k children, C_1, C_2,..., C_k. Then, for any W, $X[P, W]$ can be calculated recursively as

$$X[P, W] = w_P \lambda_P + \max_{\{W_{C_i}, t_{C_i}\}} \sum_{i=1}^{k} X[C_i, W_{C_i}] f_{C_i}(t_{C_i}), \quad (5)$$

where $\{W_{C_i}, t_{C_i}\}$ satisfy the constraints of problem π_D for node P. The intuition behind (5) is that the contribution from a child C_i that waits for a time W_{C_i} and is allowed to make t_{C_i} transmissions is $X[C_i, W_{C_i}] f_{C_i}(t_{C_i})$, and we are interested in the sum of the information.

Theorem V.2. *$X[i, W]$ computed by (5) maximizes the information received at node i if node i waits for a time W, for any node i in the tree. Consequently, $X[S, D]$ provides an optimal solution to π_D.*

The proof can be found in the appendix.

Theorem V.3. *Let C_1, C_2, ..., C_k be the children of node i that is not a leaf node. If W^* is the optimal waiting time of node i, then one of the optimal set of time slots during which the children transmit is given by $\{\max(0, W^* - \sum_{j=1}^{k} \mathcal{T}_{C_j}), \max(0, W^* - \sum_{j=1}^{k} \mathcal{T}_{C_j}) + 1, ..., W^* - 1\}$.*

Proof: We prove this theorem by contradiction.

From Theorem V.1, we know that transmitting in consecutive slots is optimal.

Suppose that the set of slots given above is not optimal. This means that at least one of the children makes a transmission before $\max(0, W^* - \sum_{j=1}^{k} \mathcal{T}_{C_j})$. Since the maximum total number of transmissions for all the nodes in the hop is given by $\sum_{j=1}^{k} \mathcal{T}_{C_j}$, no child node makes a transmission in at least one of the slots in the set above. If the child node that had transmitted before the slot $\max(0, W^* - \sum_{j=1}^{k} \mathcal{T}_{C_j})$ had waited until this free slot, it could have potentially gathered more packets, and still made a successful transmission, thus increasing $X[i, W]$. This contradicts the assumption that the above set of time slots is not optimal. ■

The implication of this theorem is that no child needs to transmit before $\max(0, W^* - \sum_{j=1}^{k} \mathcal{T}_{C_j})$. This is critical in reducing the search space for determining the optimal waiting time of each node, and will be used in the rest of the paper.

We now show that finding $X[i,W]$ for a non-leaf node i and for a given W is NP-hard in the strong sense by reducing it from the 3-partition problem [6] which is known to be NP-hard in the strong sense[1].

3-Partition Problem: Given $3m+1$ positive integers $a_1, a_2, ..., a_{3m}$, and B such that $\sum_{i=1}^{3m} a_i = mB$, and $\frac{B}{4} < a_i < \frac{B}{2}$ for $i = 1, 2, ..., 3m$; is there a partition of the set $\{1, ..., 3m\}$ into m disjoint subsets $S_1, S_2, ..., S_m$ such that $\sum_{i \in S_j} a_i = B$ for $j = 1, 2, ..., m$?

Note that if there exists a solution to this problem, each subset S_j has exactly three elements.

Theorem V.4. *Determining $X[i,W]$ for an arbitrary node i having k children, and for an arbitrary waiting time W is NP-hard in the strong sense when the input to the problem is k.*

Proof: Given an instance of the 3-partition problem, we construct the following instance of the scheduling problem with *error-free links*. Node i has $k = 3m$ children, denoted by $C_1, C_2, ..., C_{3m}$. The number of time slots required to transmit a packet over the link (C_j, i) is a_j, and the information possessed by C_j at time slot zero is also a_j, for $j = 1, 2, ..., 3m$. Further, each child C_j has $m-1$ children, and each child of C_j also accounts for a_j units of information, for $j = 1, 2, ..., 3m$. Each child of C_j takes B time slots to transmit a packet to C_j, for $j = 1, 2, ..., 3m$. Finally, we have $W = mB$ at node i. Figure 2 illustrates this construction.

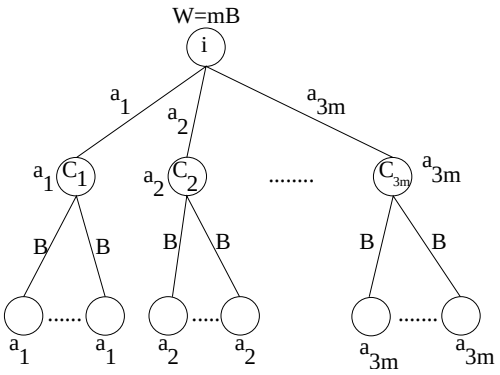


Fig. 2. Reduction from the 3-partition problem

We now show that there exists a solution to the 3-partition problem if and only if there exists a schedule such that $X[i, mB] \geq \frac{m(m+1)B}{2}$.

We show by induction that for any $j = 1, 2, ..., m$, there exists a schedule such that $X[i, jB] \geq \frac{j(j+1)B}{2}$ if and only if there exists j disjoint subsets S_1, S_2, ..., S_j of $\{a_1, ..., a_{3m}\}$, each having three elements, such that $\sum_{n \in S_l} a_n = B$ for $l = 1, 2, ..., j$.

Suppose that $j = 1$.

($\Rightarrow$): If there exists a set $S_1 = \{a_{n_1}, a_{n_2}, a_{n_3}\}$ such that $\sum_{n \in S_1} a_n = B$, then we can schedule the children C_{n_1}, C_{n_2}, and C_{n_3} in $a_{n_1} + a_{n_2} + a_{n_3} = B$ slots, and we will obtain $X[i, B] = a_{n_1} + a_{n_2} + a_{n_3} = B$.

($\Leftarrow$): To prove the converse, suppose that there exists a schedule such that $X[i, B] \geq B$. Note that for any $n < B$, $X[C_j, n] = a_j$ for $j = 1, 2, ..., 3m$. Suppose that there does not exist a set consisting of three elements such that the sum of the elements is B. Since $\frac{B}{4} < a_j < \frac{B}{2}$ for $j = 1, 2, ..., 3m$, if the schedule consists of more than three children, then the total time for all the children to transmit would exceed B slots, and if the schedule consists of less than three children, then the amount of information obtained would be strictly less than B. Hence, the schedule must exactly consist of three children. However, if the total information accounted for by these children is greater than B, then the total time to obtain this information is also greater than B (since the transmission time is identical to the information provided). Hence, we obtain a contradiction.

Thus, the result holds for $j = 1$. Assume that the result holds for $j = r$.

Consider $j = r + 1$.

($\Rightarrow$): If there exists $r+1$ disjoint subsets S_1, S_2, ..., S_{r+1} such that $\sum_{n \in S_l} a_n = B$ for $l = 1, 2, ..., r+1$, then we can schedule the children corresponding to the elements in S_l between slots $(l-1)B$ and lB, for $l = 1, 2, ..., r+1$. Since a child C_j gets an additional information of a_j after every B slots, we have $X[C_j, (l-1)B] = la_j$ for $l = 1, 2, ..., m$ and for $j = 1, 2, ..., 3m$. Hence, the children corresponding to S_l will totally account for lB units of information for $l = 1, 2, ..., r+1$. Therefore, the total information obtained by this schedule is $\frac{(r+1)(r+2)B}{2}$.

($\Leftarrow$): To prove the converse, suppose that there exists a schedule such that $X[i, (r+1)B] \geq \frac{(r+1)(r+2)B}{2}$. We can split this schedule into $r+1$ non-overlapping schedules, $S'_1, S'_2, ..., S'_{r+1}$, where the waiting time of nodes in S'_l lies between $(l-1)B$ and lB for $l = 1, 2, ..., r+1$. Therefore, if a child C_j is in schedule S'_l, then it would account for la_j units of information. Since $X[i, (r+1)B] \geq \frac{(r+1)(r+2)B}{2}$, we have

$$\sum_{l=1}^{r+1} \sum_{j: C_j \in S'_l} la_j \geq \frac{(r+1)(r+2)B}{2}. \tag{6}$$

Further, since the waiting time at node i is given by $W = (r+1)B$, we have

$$\sum_{l=1}^{r+1} \sum_{j: C_j \in S'_l} a_j \leq (r+1)B, \tag{7}$$

since it takes a_j time slots to transmit a packet from C_j to i, for $j = 1, 2, ..., 3m$.

Subtracting (7) from (6), we get

[1]In [1] and [13], we claim that the corresponding optimization problems are MAX SNP-hard, which means that unless P=NP, there does not exist any Polynomial Time Approximation Scheme (PTAS) for those problems. However, we found that the reduction to prove MAX SNP-hardness was not accurate. We correct this, and prove that these optimization problems are NP-hard in the strong sense using reduction from a 3-partition problem. Since we consider a more general problem than those considered in [1], [13], this immediately implies that the problem that we study in this work is also NP-hard in the strong sense.

$$\sum_{l=1}^{r+1} \sum_{j:C_j \in S'_l} (l-1)a_j \geq \frac{r(r+1)B}{2}. \quad (8)$$

This implies that

$$\sum_{l=2}^{r+1} \sum_{j:C_j \in S'_l} (l-1)a_j \geq \frac{r(r+1)B}{2}, \quad (9)$$

since S'_1 accounts for zero information in (9). The relation (9) implies that in rB slots, the schedule $S'_2, S'_3, ..., S'_{r+1}$ accounts for at least $\frac{r(r+1)B}{2}$ units of information. According to the induction hypothesis, this is possible if and only if the sets $S'_2, S'_3, ..., S'_{r+1}$ each contain exactly three elements, and the sum of the transmission times of the children in each set is B. Now, given that $S'_2, S'_3, ..., S'_{r+1}$ are disjoint subsets each containing three elements whose sum is B, $X[i,(r+1)B] \geq \frac{(r+1)(r+2)B}{2}$ if and only if S'_1 also consists of three children such that the sum of their transmission times is B.

Hence, the result follows by induction, and therefore, determining $X[i,W]$ is NP-hard in the strong sense when the input to the problem is the number of children of i. ■

This result implies that even if the transmission times, and the deadline are bounded by a polynomial in the input, the problem remains NP-hard. Since the deadline is typically linear in the input, it is important to show NP-hardness in the strong sense for this problem. This means that a pseudo-polynomial time optimal algorithm cannot exist for this problem unless P=NP.

Relationship to the Job Interval Selection Problem: We now show that computing $X[i,W]$ is a Job Interval Selection Problem (JISP) in a single machine. JISP is a well-studied problem in the machine scheduling literature [17], [18], [19]. It is known to be MAX SNP-hard, which implies that unless P=NP it is not possible to find a Polynomial-Time Approximate Solution (PTAS) to JISP.

We briefly describe the Job Interval Selection Problem (JISP). In a single-server JISP, n jobs need to be served by a single machine. Each job has k instances, where each instance is associated with an explicit time interval during which it must be scheduled, and a certain profit. The machine can only serve one instance of any job during each time slot. The goal is to find a schedule such that at most one instance of a job is present in the schedule, the instances in the schedule do not conflict in time, and the sum of the profits of the job instances is maximum.

Determining $X[i,W]$ for an arbitrary node i and an arbitrary waiting time W is a JISP. This can be seen as follows. In our problem, the jobs correspond to the children nodes that need to be served by the parent. Each instance of the job (child) corresponds to the interval during which it transmits, and a weight. The interval corresponds to the interval of the job instance, and the node weight corresponds to the profit of the job instance. Thus, our problem is a JISP.

There exists $\frac{1}{2}$-approximation algorithms for special cases of JISP. However, JISP is a relatively small part of our problem. While finding $X[i,W]$ for each node i and for each W is a JISP, we ultimately need $X[S,D]$. If we use existing JISP approximation algorithms, they could result in a very poor performance (since if there are h hops in the tree, the approximation factor of the overall problem could be as poor as $\frac{1}{2^h}$). Further, the input to our problem is the maximum node degree of the tree, which is typically small. Therefore, we take these issues into account, and develop a new solution.

VI. An Optimal Solution to π_D

In this section, we propose an optimal algorithm to the interval scheduling sub-problem, i.e., to compute $X[i,W]$ for a non-leaf node i and for any waiting time $W \in \{0,1,...,D-1\}$. While the problem is strongly NP-hard in general, we obtain an algorithm of time complexity $O(k^3 2^k)$, where k denotes the maximum node degree in the data aggregation tree. Hence when $k = O(\log N)$, an optimal schedule at each node can be found in polynomial time, which, together with the dynamic programming framework proposed in the previous section, provides an optimal polynomial time algorithm to the deadline constrained optimization problem. For large k, we further propose a sub-optimal algorithm having a low computational complexity in the next section.

The interval scheduling subproblem that needs to be solved can be formally defined as follows.

Single-machine scheduling with a common deadline and multiple time windows: Let $\mathcal{J} = \{J_1, ..., J_k\}$ denote a set of k jobs. Each job J_i can be processed in one of m_i time windows, $\{(s^i_1, l^i_1, u^i_1), ..., (s^i_{m_i}, l^i_{m_i}, u^i_{m_i})\}$, where s^i_j, l^i_j, u^i_j are integers, and s^i_j denotes the j-th release time (the time at which the job is released to the machine), l^i_j denotes the j-th processing time, and u^i_j denotes the weight (or profit) if J_i is scheduled in the time interval $[s^i_j, s^i_j + l^i_j]$. Given an integer W, a feasible schedule is a subset of time windows, such that all the intervals are subsets of the interval $[0,W]$, at most one time window is selected for each job, and no two intervals in the schedule overlap. A schedule is optimal if the total weight of the scheduled jobs is maximized.

We note that when each job only has a single time window associated ($m_i = 1 \ \forall i$), and a common deadline W, there exists an optimal dynamic programming based solution with polynomial time complexity in k and W [20]. We propose the following extension to solve the general problem. First consider the collection of all the time windows belonging to all the jobs, and sort them in non-increasing order of release times. Ties are broken arbitrarily. Denote the sorted time windows as $\mathbb{T} = \{T_1, ..., T_n\}$, where $n = \sum_{j=1}^{k} m_j$, $T_i = (v_i, s_i, l_i, u_i)$, where v_i, s_i, l_i, u_i denote the index of the job, the release time, the processing time, and the weight of the i-th time window, respectively. Let $Z[i, \mathcal{I}, s]$ denote the maximum weight that can be achieved by a feasible subset of time windows $\mathbb{T}' \subseteq \mathbb{T}$, such that (1) $\mathbb{T}' \subseteq \{T_1, ..., T_i\}$; (2) the set of jobs corresponding to time windows in $\mathbb{T}'$ is $\mathcal{I} \subseteq \mathcal{J}$; (3) $\mathbb{T}'$ can be scheduled within time interval $[s,W]$ subject to the release time constraints. Finally, the optimal solution is the maximum value of $Z[n, \mathcal{I}, s]$ for $\mathcal{I} \subseteq \mathcal{J}, s \in \{0, ..., W\}$.

The algorithm first initializes $Z[i, \mathcal{I}, s]$ to zero for all $i = 0, ..., n, \mathcal{I} \subseteq \mathcal{J}$ and $s = 0, ..., W$. The algorithm then proceeds by checking each T_i from $i = 1$ to n. Consider the i-th round.

The algorithm iterates through each pair of $(\mathcal{I}, s)$, and sets $Z[i,\mathcal{I},s] = \max(Z[i-1,\mathcal{I},s], Z[i-1,\mathcal{I}\backslash\{v_i\}, s+l_i]+u_i)$ if $v_i \in \mathcal{I}$, $s \geq s_i$, and $s+l_i \leq W$. Otherwise, it sets $Z[i,\mathcal{I},s] = Z[i-1,\mathcal{I},s]$. The main steps of the algorithm are as follows.

Algorithm 1 Computing $X[i,W]$ for any non-leaf node i and waiting time W for problem Π_D

Sort the time windows, denoted as $\{T_i = (v_i, s_i, l_i, u_i)\}_{i=1,...,n}$, in non-increasing order of start times s_i.
for $i = 0 \rightarrow n$ **do**
  for $\mathcal{I} \subseteq \mathcal{J}$ **do**
    for $s = 0 \rightarrow W$ **do**
      $Z[i,\mathcal{I},s] \leftarrow 0$
for $i = 1 \rightarrow n$ **do**
  for each pair of $(\mathcal{I}, s)$ such that $\mathcal{I} \subseteq \mathcal{J}$ and $s \leq W$ **do**
    if $v_i \notin \mathcal{I}$ OR $s < s_i$ OR $s + l_i > W$ **then** $Z[i,\mathcal{I},s] \leftarrow Z[i-1,\mathcal{I},s]$
    else $Z[i,\mathcal{I},s] \leftarrow \max(Z[i-1,\mathcal{I},s], Z[i-1,\mathcal{I}\backslash\{v_i\}, s+l_i]+u_i)$
return $\max_{\mathcal{I}\subseteq\mathcal{J}, s\in\{0,...,W\}} Z[n,\mathcal{I},s]$

We now prove the correctness of Algorithm 1.

Theorem VI.1. *Algorithm 1 is optimal for computing $X[i,W]$ for any non-leaf node i and waiting time W.*

The proof can be found in the appendix.

We now analyze the time complexity of Algorithm 1. It takes $O(n \log n)$ time to sort the time windows, where $n = \sum_{j=1}^{k} m_j$. If $m = \max_{1\leq j\leq k} m_j$, then $n = O(mk)$. The loop runs n times, and in each round, $2^k W$ pairs of $(\mathcal{I}, s)$ are examined. For each pair of $(\mathcal{I}, s)$, it takes constant time to update $Z[i,\mathcal{I},s]$. Hence the time complexity of the algorithm is $O(mk \log(mk) + mkW2^k)$.

When applied to our problem, k is bounded by the maximum number of children of any node in the tree. For a given $W \in \{0,...,D-1\}$, m is bounded by $\min(k\mathcal{T}, W)\mathcal{T}$, where $\mathcal{T} = \max_{i,j} \mathcal{T}_{i,j}$ is a constant, since for any children, it suffices to consider the waiting times in $\{\max(W-k\mathcal{T},0),...,W\}$, and for each waiting time, at most $\mathcal{T}$ time slots will be allocated for transmission. Furthermore, since a feasible schedule takes at most $\min(k\mathcal{T}, W)$ time slots, it suffices to compute $Z[i,\mathcal{I},s]$ for $s \in \{\max(W-k\mathcal{T},0),...,W\}$. This implies that it suffices to compute $Z[i,\mathcal{I},s]$ for only $\min(k\mathcal{T}, W)$ values of s. Since $\mathcal{T}$ is a constant, $m = O(k)$. Hence, the time complexity of computing $Z[\cdot,\cdot,\cdot]$ is $O(k^2 \log(k) + k^3 2^k) = O(k^3 2^k)$. Thus, the running time of the algorithm at a node having k children is $O(k^3 2^k)$ for any $W \in \{0,...,D-1\}$. This implies that the complexity of computing $X[i,\cdot]$ at any node i is $O(Dk^3 2^k)$.

Note that Algorithm 1 only computes $X[i,W]$ for a given node i, and a given waiting time W. We ultimately need to calculate $X[S,D]$ at the sink to solve Π_D. We propose the following dynamic programming based algorithmic framework for this purpose.

A. Algorithmic framework to calculate $X[S,D]$

Algorithm 2 Computing optimal solution for problem Π_D

for each leaf node $i \in V_L$ **do**
  for $W = 0 \rightarrow D-1$ **do**
    $X[i,W] = w_i \lambda_i$
for each non-leaf node i such that $X[\cdot,\cdot]$ has been computed for all of i's children **do**
  for $W = 0 \rightarrow D-1$ **do**
    Compute $X[i,W]$ using Algorithm 1
Finally, compute $X[S,D]$ at the root
Look up $X[S,D]$, and assign optimal waiting times, and transmissions slots to the root's children
Go down the tree from the root to the leaves assigning optimal waiting times and transmission slots at each node by looking up the corresponding vector $X[\cdot,\cdot]$.

Algorithm 2 is optimal because of the correctness of the recursion to compute $X[\cdot,\cdot]$ (Theorem V.2). Using this algorithmic framework, the overall complexity of computing an optimal solution to problem Π_D is $O(hDk^3 2^k)$, where h is the maximum number of hops in the tree (assuming that nodes at the same height can perform computations in parallel). We will use this framework for all the other algorithms developed in this work as well.

VII. Sub-optimal Formulation and Solution to π_D

In this section, we propose a sub-optimal solution to the interval scheduling subproblem for large k, which, together with the dynamic programming framework proposed in Section V, provides an efficient solution to the deadline constrained problem. The structural properties of π_D imply that in any hop the next node starts transmitting only after the previous node has finished transmitting. This is because the optimal schedule for each child in that hop is an interval, and the schedules of any two children cannot conflict, i.e., no two intervals can intersect. Therefore, in each hop, there is an order in which children are allocated time slots for transmission. The idea of the sub-optimal formulation is that we assume that in any hop, the order in which children transmit to their parent is known. We denote this problem Π_D^{sub}.

To be precise, let the symbol "$\rightarrow$" represent the order of transmission. For instance, $C_1 \rightarrow C_2$ means that C_1 is scheduled for transmissions before C_2. For each non-leaf node i in the network (including the sink), let the set of children $C(i) = \{i_1, i_2, ..., i_{|C(i)|}\}$, and the order of transmission be given by $i_1 \rightarrow i_2 \rightarrow ... \rightarrow i_{|C(i)|}$. From Theorem V.1 and Theorem V.3, we know that transmitting in consecutive time slots is optimal, and that there are no unscheduled slots once the first child starts transmitting. Also, $W_{i_1} \leq W_{i_2} \leq ... \leq W_{i_{|C(i)|}}$). It follows that the constraint (2) in Π_D can be replaced by the following constraint. For each $i \in V \cup \{S\}\backslash V_L$,

$$\sum_{j:j\in C(i)} t_j \leq W_i - W_{i_1}. \tag{10}$$

Thus, the only difference between problem Π_D and problem Π_D^{sub} is that the constraint (2) is replaced by (10). It turns out that once the order of transmission in each hop is known, the resulting problem can then be solved in polynomial time. Hence our sub-optimal solution to the interval scheduling subproblem has two steps. First, we fix the order in which children transmit to their parent in each hop. We can design intelligent heuristics for fixing such an order. We design such a heuristic to evaluate our sub-optimal solution in Section IX. Second, for this given order for transmission, we solve the problem Π_D^{sub}, which is the focus of this section.

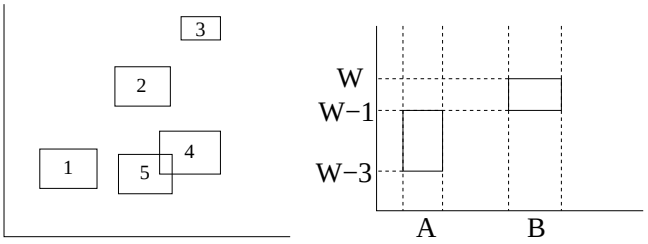
Note that even if we fix the order, we still need to determine the waiting times and the number of time slots allocated to each node. For instance, consider two children C_1 and C_2. They have waiting times W_1 and W_2, respectively, and are allocated t_1 and t_2 time slots, respectively. Suppose we know that C_1's schedule is *before* C_2's schedule. Then, from Theorem V.1, we know that $W_1 + t_1 - 1 < W_2$. We now need to determine W_1, t_1, W_2, and t_2, with the additional constraint that the order in which children are scheduled is known.

We now provide some graph theoretic preliminaries required to solve Π_D^{sub}.

A. Preliminaries

1) **Maximum Weight Independent Set:** An Independent Set in a graph $G(V, E)$ is a set of vertices $U \subseteq V$ such that no two vertices in U have an edge between them. A Maximum Weight Independent Set in G is an Independent Set of maximum total weight of vertices.
2) **Interval graph:** Let $\{I_1, I_2, ..., I_n\}$ be a set of intervals on the real line. Then, the interval graph $G(V, E)$ corresponding to this set of intervals is defined as follows.
   - $V = \{I_1, I_2, ..., I_n\}$. Each vertex denotes an interval.
   - For any $y, z \in \{1, 2, .., , n\}$, $(I_y, I_z) \in E$ if and only if the intervals intersect, i.e., $I_y \cap I_z \neq \emptyset$.
3) **Interval graph of interval number m:** The definition is identical to that of the interval graph except that each vertex can now be represented as a disjoint union of m intervals. An interval graph of interval number 2 is called a *double interval graph*.
4) **Rectangle graphs:** Rectangle graphs are a subclass of double interval graphs. A double interval graph can be transformed into a rectangle graph by simply labeling the vertices in the double interval graph as the set-product of the two intervals instead of the union of the two intervals. Thus, each vertex now represents a rectangle in $\mathbb{R}^2$. It is important to note that two rectangles that do not intersect need not form an independent set in the corresponding double interval graph. On the other hand, every independent set in the double interval graph is an independent set in the rectangle graph.
5) Maximum Weight Independent Set on interval graphs (order 1) can be found in polynomial time. However, Maximum Weight Independent Set on interval graphs of order m, $m > 1$, is still NP-hard [21].
6) **Increasing Independent Set on rectangle graphs:** An Increasing Independent Set on a rectangle graph is an independent set that has the following property.
   Let $A = \{r_1, r_2, ..., r_m\}$ be an ordered set of rectangles ordered in the following fashion. Any $i, j \in \{1, 2, ..., m\}$ such that $i < j$ must obey the following.
   - The maximum x-coordinate of any point in r_i is at most equal to the minimum x-coordinate of any point in r_j.
   - The maximum y-coordinate of any point in r_i is at most equal to the minimum y-coordinate of any point in r_j.

   Then, A is an Increasing Independent Set on the given rectangle graph. Note that the rectangles in A are ordered such that the next rectangle is *to the right and to the top* of the previous rectangle in the order. Further, an increasing independent set on a rectangle graph is an independent set on the corresponding double interval graph.



(a) Rectangle Graphs (b) Increasing Independent Set

Fig. 3. Rectangle Graphs and Independent Sets

We now provide an example to illustrate the definitions above. Consider Figure 3(a). Each rectangle in the figure represents a vertex in a rectangle graph. There will be an edge between two vertices only if the corresponding two rectangles intersect. For instance, there is an edge between rectangles 4 and 5. $\{1, 2, 3, 4\}$ and $\{1, 2, 3, 5\}$ are two maximal independent sets of rectangles. While $\{1, 4\}$ is an independent set in the rectangle graph, it does not form an independent set in the corresponding double interval graph because its intervals on the y-axis intersect. Also, $\{1, 2, 3\}$ is an example of an Increasing Independent Set in the rectangle graph because rectangle 2 is to the right and to the top of rectangle 1, and rectangle 3 is to the right and to the top of rectangle 2.

Figure 3(b) shows an Increasing Independent Set for a node with two children A and B, where A has to make two transmissions, and B has to make one transmission before a deadline W. Assuming that A transmits before B, an Increasing Independent Set is given by A transmitting from $W - 3$ to $W - 1$, and B transmitting from $W - 1$ to W. Clearly, the rectangle for B is to the top and to the right of the rectangle for A.

B. Solution

We first construct a graph, G', as follows. For each C_j, $j \in \{1, 2, ..., k\}$, for each t_{C_j}, $t_{C_j} \in \{0, 1, 2, ..., \mathcal{T}_{C_j}\}$,

construct nodes labeled $(C_j, W - \sum_{j=1}^{k} \mathcal{T}_{C_j}, t_{C_j})$, $(C_j, W - \sum_{j=1}^{k} \mathcal{T}_{C_j} + 1, t_{C_j})$, ..., $(C_j, W - t_{C_j}, t_{C_j})$, respectively. The first term in the label represents the child, the second term represents the waiting time of the child, and the third term represents the number of transmissions made by the child. A node labeled (C_j, W_{C_j}, t_{C_j}) in this graph is assigned a weight $X[C_j, W_{C_j}](f_{C_j}(t_{C_j}))$. For any two nodes (C_i, W_i, t_i) and (C_j, W_j, t_j) in G', there exists an edge if and only if *(a)* $\{W_i, W_i+1, ..., W_i+t_i-1\} \cap \{W_j, W_j+1, ..., W_j+t_j-1\} \neq \emptyset$, or *(b)* $C_i = C_j$.

We show that when the order of schedules of children in any hop of the tree is known, finding $X[\cdot, \cdot]$ is equivalent to finding a Maximum Weighted Increasing Independent Set in the rectangle graph corresponding to G'.

Theorem VII.1. *Consider any hop with parent i having k children C_1, ..., C_k. WLOG, assume that the order in which the children are scheduled is $C_1 \rightarrow C_2 \rightarrow ... \rightarrow C_k$. Assign an interval $[a_i, b_i]$ to child C_i, $i \in \{1, 2, ..., k\}$, such that $W < a_1 < b_1 < a_2 < b_2 < ... < a_k < b_k$, and replace the first term in the label of each node in G' by this interval. Then, G' is a double interval graph, and $X[i, W]$ can be obtained by finding a Maximum Weighted Increasing Independent Set in the rectangle graph corresponding to G'.*

Proof: Assign an interval $[a_i, b_i]$ to child C_i, $i \in \{1, 2, ..., k\}$, such that $W < a_1 < b_1 < a_2 < b_2 < ... < a_k < b_k$. Replace C_i by $[a_i, b_i]$ in the first term of the label of each node in G'. Note that the second and the third terms in the label of each node of G' is an interval which specifies the time slots during which the child transmits. Each node in G' can now be represented as the union of two disjoint intervals, the first interval corresponding to the child, and the second interval corresponding to the time slots during which it transmits. Further, for any two nodes in G' that represent the same child, there exists an edge between the two nodes since the first interval in both the node labels have a non-empty intersection. Therefore, G' is a double interval graph.

G' can now be transformed into a rectangle graph as defined before. Let the x-axis represent the children, and the y-axis represent the schedules. Consider any two children, C_l and C_m. Let $l < m$, and hence C_l transmits before C_m. In the rectangle graph, a non-conflicting schedule for C_l and C_m can be achieved if and only if the rectangle corresponding to C_m's schedule is to the top and to the right of the rectangle corresponding to C_l's schedule. It cannot be to the left because $a_m > b_l$. It cannot be to the bottom because that would contradict the assumption that C_l transmits before C_m.

Thus, $X[i, W]$ can be obtained by finding a Maximum Weighted Increasing Independent Set in this rectangle graph corresponding to G'. ∎

In [22], an algorithm has been proposed to determine a Maximum Weighted Increasing Independent Set in a rectangle graph for determining similarities in DNA sequences. For n rectangles in the rectangle graph, the complexity of this algorithm is $O(n \log n)$. G' has $O(k^2)$ vertices. Therefore, the complexity of finding $X[i, W]$ in Theorem VII.1 is $O(k^2 \log k)$.

Algorithm 3 summarizes how to compute $X[i, W]$ for any non-leaf node i, and waiting time W.

Algorithm 3 Computing $X[i, W]$ for problem Π_D^{sub}

1: Construct graph G' as shown in Section VII-B
2: Construct the rectangle graph corresponding to G' as showin in Theorem VII.1
3: Find a Maximum Weight Increasing Independent Set in the rectangle graph corresponding to G'

We can now include this interval scheduling algorithm in the algorithmic framework (Algorithm 2) developed in Section VI-A to calculate an optimal solution to Π_D^{sub}.

Theorem VII.2. *Algorithm 3 provides a collision-free schedule, and accurately determines $X[i, W]$ for any non-leaf node i and waiting time W.*

Proof: This result follows from the previous results in the paper. ∎

The computational complexity of determining an optimal solution to Π_D^{sub} for a tree with a maximum of h hops where each hop has k children on average can easily be calculated to be $O(hDk^2 \log k)$, where D is the deadline. One can thus see that the sub-optimal version has a very low computational complexity.

To summarize, we make the following observations.

- Problem Π_D is NP-hard in the strong sense. However, in our problem, the exponential complexity is in the number of children (k) in a hop.
- If k is small (which is typically the case), then with $O(hDk^3 2^k)$ complexity we can solve problem Π_D. Otherwise, we can use the sub-optimal version that has a complexity $O(hDk^2 \log k)$.

C. Discussion

In certain tree structures, the order in which children are scheduled does not affect the optimal solution of problem Π_D.

1) **Single hop tree networks:** Since all nodes apart from the sink are leaf nodes, only the sink performs in-network computation. Therefore, the order in which leaf nodes are scheduled does not matter. Hence, the sub-optimal solution is optimal in this case.
2) **Symmetric trees:** A symmetric tree is defined as one in which nodes that are equal number of hops away from the sink satisfy the following:
   a) They have equal number of children.
   b) Either all of them are source nodes or none of them are source nodes, and they provide the same amount of information.
   c) Each incoming link has the same link reliability function.

   An example of a symmetric tree is given in Figure 4(a). It is easy to see that the order of transmission of children

in any hop of a symmetric tree does not affect the optimal solution to problem Π_D.

3) We now combine the ideas from the above two cases to construct our last example. Consider a tree that is symmetric, except at the farthest hop from the sink. Specifically, we relax condition (c) at parents of leaf nodes. Incoming links of a node, whose children are all leaf nodes, need not have the same link reliability function. However, the link reliability functions must be identical across hops. For example, consider two nodes, P and Q, with two children each. Both these children are leaf nodes. If the link reliability functions for the incoming links of P be f_1 and f_2, then the link error probabilities for the incoming links of Q must also be f_1 and f_2. However, f_1 need not be identical to f_2, unlike the symmetric case. It can be easily seen that the sub-optimal algorithm is optimal in this case as well.

Since the only difference between the sub-optimal and the optimal problem is in determining the order in which nodes transmit in a hop, one can use intelligent heuristics to come up with a particular policy for scheduling. However, in general, choosing the optimal order of transmission in any hop depends on a number of factors such as the number of source nodes in the sub-tree, the entire structure of the sub-tree, and the link errors in the sub-tree.

VIII. Solution to Π_{DE}

Having completely studied Π_D, we can now proceed to developing a solution for the original problem Π_{DE} that we formulated in Section IV. The idea is to use the structure that we developed for Π_D to compute the optimal solution for Π_{DE}. We also develop a sub-optimal version similar to Π_D^{sub}, and solve it by computing an energy constrained Maximum Weight Increasing Independent Set.

Similar to solving Π_D, we can solve Π_{DE} in a recursive manner.

Let $X[i,W,r]$ be the information obtained at i from the sub-tree rooted at node i if i waits for a time W, and consumes at most r units of energy on receptions. We can define $X[i,W,r]$ recursively as follows. For any leaf node l, $X[l,W,r] = w_l\lambda_l$. For a non-leaf node i, $X[i,W,r] = w_i\lambda_i + \max_Q \sum_{j\in C(i)} X[j,W_j,\lfloor\frac{E_j - e_jE_T}{E_R}\rfloor]f(t_j,e_j)$, where Q is the set of constraints including (2) for node i, and the constraint $\sum_{j\in C(i)} e_j \le r$. The first term represents the information from i, and the second term represents the information from i's children to i. Note that if child j allows e_j units of energy for transmissions, it can allow at most $\lfloor\frac{E_j-e_jE_T}{E_R}\rfloor$ units of energy on receptions. This is represented in the second term.

We can show that by computing $X[S,D,\lfloor\frac{E_S}{E_R}\rfloor]$, we can obtain an optimal solution to problem Π_{DE}.

Theorem VIII.1. *$X[i,W,r]$ maximizes the information that can be obtained at i from the sub-tree rooted at node i if i waits for a time W, and consumes at most r units of energy on receptions. Consequently, $X[S,D,\lfloor\frac{E_S}{E_R}\rfloor]$ provides an optimal solution to problem Π_{DE}.*

Proof: The proof follows by induction, and is similar to that of Theorem V.2. ■

A. Optimal solution to Π_{DE}

We now extend Algorithm 1 to take the extra energy constraint at each node into account. To compute $X[i,W,r]$, we need to solve the following extension of the interval scheduling subproblem defined in Section VI, where each job J_i is again associated with m_i time windows with each time window having the following form $(s_j^i, l_j^i, e_j^i, u_j^i)$, where the new symbol e_j^i denotes the amount of energy allocated. Further, for any feasible solution, the total energy allocated to the time windows selected must be bounded by r.

We extend our solution to Π_D as follows. First, we again obtain the sorted collection of time windows, represented by $\mathbb{T} = \{T_1, ..., T_n\}$, by sorting according to non-increasing order of release times s_j^i. Here, $T_i = (v_i, s_i, l_i, e_i, u_i)$. Let $Z[i,\mathcal{I},s,e]$ denote the maximum weight that can be achieved by a feasible subset of time windows $\mathbb{T}' \subseteq \mathbb{T}$, such that it satisfies (1)-(3) defined in Section VI, and the following additional condition: (4) the total energy allocated to $\mathbb{T}'$ is at most e. The algorithm initializes $Z[i,\mathcal{I},s,e]$ to zero for each $i = 0, ..., n, \mathcal{I} \subseteq \mathcal{J}, s = 0, ..., W-1$, and $e = 0, ..., r$. The algorithm then proceeds by checking each T_i from $i = 1$ to n. Consider the i-th iteration. The algorithm iterates over each triplet $(\mathcal{I}, s, e)$, and sets $Z[i,\mathcal{I},s,e]$ to be the maximum of $Z[i-1,\mathcal{I},s,e]$ and $Z[i-1,\mathcal{I}\backslash\{v_i\}, s+l_i, e-e_i] + u_i$ when $v_i \in \mathcal{I}$, $s + l_i \le W$, $s_i \ge s$, and $e \ge e_i$. Otherwise, it sets $Z[i,\mathcal{I},s,e] = Z[i-1,\mathcal{I},s,e]$.

The correctness of the algorithm can be proved in a similar way as that for Algorithm 1. The time complexity is given by $O(mk\log mk + mkWr2^k)$ since we also need to iterate over r values of e for each iteration. When applied to a node i in our problem, we have $r \in \{0, ..., \lfloor E_i/E_R\rfloor\}$, and $e_j^i \in \{0, ..., \mathcal{E}_i\}$, where $\mathcal{E}_i$ is assumed to be a constant. For a given W, m is bounded by $\min(k\mathcal{T}, W)\mathcal{T}\mathcal{E}$, where $\mathcal{E} = \max_i \mathcal{E}_i$. Furthermore, it suffices to compute only $Z[i,\mathcal{I},s,e]$ for $e \in \{1, ..., \min(k\mathcal{E}, r)\}$ for a given r. Hence, the time complexity of the algorithm for a given W and r is $O(k^4 2^k)$.

By again appropriately incorporating the algorithmic framework for Algorithm 2 in Section VI-A, we can obtain an optimal solution to Π_{DE}. The complexity of this optimal algorithm is $O(hDr_{max}k^4 2^k)$, where r_{max} is the maximum units of energy allowed for receptions at any node in the network.

B. Sub-optimal solution to Π_{DE}

We now develop a sub-optimal algorithm that computes $X[i,W,r]$ for a non-leaf node i in the tree, assuming that the order of transmission for its children is given. As mentioned before, the idea is to find an energy constrained Maximum Weight Increasing Independent Set.

Suppose that i has k children, $1, 2, ..., k$. WLOG, assume that the order in which the children transmit is given by $1 \to 2 \to ... \to k$, i.e., child m transmits before child n if $m < n$. We define an *Increasing Independent Set starting from a specific rectangle* as one that does not include any rectangles that are to the left or to the bottom of the given rectangle.

For instance, in Figure 3(a), an Increasing Independent Set starting from rectangle 1 is $\{1, 2, 3\}$, whereas an Increasing Independent Set starting from rectangle 2 is $\{2, 3\}$.

Let (j, w, t) represent the rectangle corresponding to child j when it waits for a time w and uses t time slots for transmission. Let $m[j, e, w, t]$ represent the weight of the Maximum Weight Increasing Independent Set starting from the rectangle (j, w, t) that expends at most e units of energy, in total, on transmissions. We define $m[j, e, w, t]$ recursively, starting from child k, as follows.

$$\begin{aligned} m[k, e, w, t] = \quad & X[k, w, \lfloor \frac{E_k - eE_T}{E_R} \rfloor] f(t, e), e \leq r \\ m[j, e, w, t] = \quad & \max_{e' \leq e} (X[j, w, \lfloor \frac{E_j - e'E_T}{E_R} \rfloor] f(t, e') + \\ & \max_{(w', t') \in Q'} m[j + 1, e - e', w', t']), \\ & j < k, e \leq r, \end{aligned} \tag{11}$$

where Q' is the set of constraints including $w + t \leq w' \leq W$, and $t' \leq \min(W - w - t, \mathcal{T}_{j+1})$.

The meaning of $m[j, e, w, t]$ in (11) is the following. The first term represents the contribution from rectangle (j, w, t) when child j expends e' units of energy on transmission. If j expends e' units of energy on transmissions, then the children $j + 1, j + 2, ..., k$ can at most expend $e - e'$ units of energy, in total, on transmissions. Further, since j starts transmitting at w, and transmits for t slots, the next child can only start transmitting after $w + t$ slots. Hence, we consider the Maximum Weight Increasing Independent Sets starting from child $j + 1$ when child $j + 1$ starts transmitting after $w + t$ slots, and the children $j + 1, j + 2, ..., k$ expend at most $e - e'$ units of energy on transmissions.

We now prove the correctness of (11).

Theorem VIII.2. *Consider any node i with k children $1, 2, ..., k$. $m[j, e, w, t]$ computed by (11) is the maximum information that reaches node i from children $j, j + 1, ..., k$, when the child j has a waiting time w and transmits for t slots, and node i expends at most e units of energy on receptions from $j, j + 1, ..., k$. Consequently, $X[i, W, r] = \max(m[1, r, \cdot, \cdot])$.*

The proof can be found in the appendix.

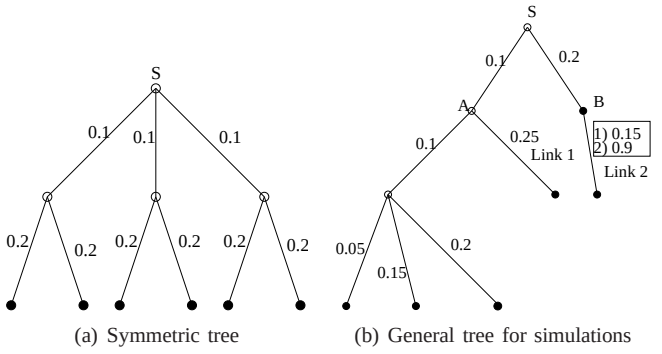
Thus, (11) provides a way to compute $X[i, W, r]$ for a given order of transmission of i's children. Using the algorithmic framework in Section VI-A, we can determine $X[S, D, \lfloor \frac{E_S}{E_R} \rfloor]$ for a given order of transmissions at each hop.

Finally, we study the computational complexity of this algorithm. Let h be the maximum number of hops in the tree, k be the maximum node degree, and r_{max} be the maximum units of energy allowed for receptions at any node in the network. We first analyze the complexity of computing $X[i, W, r]$ for a non-leaf node i, for a given waiting time W, reception energy constraint r, and a given order of transmission of i's children. Recall that $X[i, W, r]$ is computed using (11). There are $O(k^2)$ recursions involved in (11) since there are $O(k^2)$ rectangles in the rectangle graph in problem Π_D. Each recursion has a complexity of $O(kr)$ since the first maximum in $m[j, e, w, t]$ in (11) involves $O(r)$ elements, and the second maximum involves $O(k)$ elements. Therefore, the complexity of computing $X[i, W, r]$ for a given W, r, and order of transmission of children is $O(k^3 r)$. Therefore, the overall complexity is $O(hDr_{max}^2 k^3)$.

We infer that the scheduling problem involving deadline constraints is indeed the computationally complex part of problem Π_{DE}. In fact, the problem has a very low computational complexity if there were only energy constraints [14]. However, we note that we still have a distributed optimal solution, and that the exponential complexity in the deadline constrained problem is only in the maximum node degree of the tree.

IX. Numerical Results

In this section, we wish to numerically study how the ordering of transmissions in each hop affects the overall performance. Since energy constraints do not have an impact in this aspect, we only consider problem Π_D. We have provided numerical results considering energy constraints in [15]. We numerically investigate the performance of the sub-optimal solution with the optimal solution for general trees for Π_D. For the purpose of ordering nodes in the sub-optimal solution, we order nodes (having the same parent) to transmit such that a node with a greater number of source nodes in its sub-tree transmits later than a node with a lesser number of source nodes in its sub-tree. We call this heuristic H. We use retransmissions as the error-recovery scheme for these simulations. The link reliability of the link from a node i to its parent is given by $f_i(t_i) = (1 - p_i^{t_i})$, where p_i is the probability of error over link $(i, P(i))$.



(a) Symmetric tree (b) General tree for simulations

Fig. 4. Symmetric and general trees

We consider the tree in Figure 4(b) with link error probabilities as shown and source nodes represented by filled circles. We consider two trees with the same structure but with one of the links (Link 2 in Figure 4(b)) having a different probability of error (shown in a box in Figure 4(b)). The deadline is varied from 5 to 75. For each link $(i, P(i))$, we select $\mathcal{T}_i$ such that the probability that a packet gets lost in all the $\mathcal{T}_i$ transmissions is less than 0.01.

We first set the probability of error of Link 2 to be 0.15. From Figure 5(a), we can see that heuristic H is optimal for this tree with the given link error probabilities. Note that the JISP approximation algorithm in [17] performs very poorly

relative to our algorithms. This is because JISP is only a part of our problem, and these algorithms cannot be directly applied to our problem since we have a multi-hop network.

We now change the probability of error of Link 2 to 0.9. Now from Figure 5(b), we observe that heuristic H is actually not optimal for certain deadlines. This is because for these deadlines the optimal solution is for B to transmit before A. However, when the deadline is small or large, the sub-optimal solution is optimal. This can be reasoned as follows. When the deadline is small (less than 10 time slots), the amount of information that is obtained from node B is not much since B accounts for only two nodes. Since A accounts for four nodes, it is optimal for A to wait longer than B to gather packets from its predecessors, and transmit after B finishes transmitting. However, when the deadline is slightly larger, A would have gathered packets by a certain time slot, which occurs much before the deadline. However, since the link from B's child has a high probability of error, B might not have gathered its child's packet. Therefore, if A was scheduled before B, B could have waited longer to give more time to its child to transmit its packet. Therefore, the optimal order of transmission here is to transmit B after A. Finally, when the deadline is very large, the order of scheduling of A and B no longer matters because each have sufficient time to gather packets from their predecessors. Note that we do not show the JISP algorithm in Figure 5(b) since the JISP provides an information less than 2 units, and the difference between the sub-optimal and the optimal solution is very small that the difference cannot be observed if the JISP algorithm is included.

Furthermore, we observe that even if some links are bad, the heuristic can still be close to optimal. For this experiment, we vary the error probability of Link 1 from 0.05 to 0.95 while keeping that of Link 2 fixed, and vice versa. We fix the deadline to be 15 time slots. From Figure 5(c), it is clear that irrespective of the error probability of Link 1, the sub-optimal solution is still optimal. However, for Link 2, as the error probability increases above 0.35, the sub-optimal solution begins to deviate from the optimal solution. Therefore, high error probabilities do not necessarily change the optimal order of transmission.

X. Conclusion

In this paper, we have studied the problem of maximizing information in tree networks with unreliable links in the presence of deadline and energy constraints. We formulated an integer programming problem that explicitly accounted for interference, link errors, per-sensor energy constraints, and deadlines. We first studied the problem without energy constraints, showed that the integer programming problem was NP-hard in the strong sense, and looked at a sub-optimal version. We provided a low complexity, distributed optimal solution to the sub-optimal version, and analyzed tree structures for which the sub-optimal solution was actually optimal. We used the insights obtained for the deadline constrained problem to develop a distributed optimal algorithm based on dynamic programming for the original problem (including energy constraints). Further, we studied the performance of our algorithm for arbitrary tree structures through simulations, and understood when it performs optimally and when it does not. Future work involves incorporating aspects such as ARQ instead of simply allocating a fixed number of slots, or a fixed amount of energy to each link. These problems are significantly more challenging as they involve a decision process based approach, and become computationally complex even for simple tree structures.

References

[1] S. Hariharan and N. B. Shroff, "Deadline constrained scheduling for data aggregation in unreliable sensor networks," in *International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WIOPT)*, 2011.

[2] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient Communication Protocols for Wireless Microsensor Networks," in *International Conference on System Sciences*, 2000.

[3] Y. Wu, S. Fahmy, and N. B. Shroff, "On the construction of a maximum-lifetime data gathering tree in sensor networks: Np-completeness and approximation algorithm," in *IEEE INFOCOM*, 2008.

[4] A. Goel and D. Estrin, "Simultaneous optimization for concave costs: Single sink aggregation or single source buy-at-bulk," in *SODA*, 2003.

[5] H. L. V. Trees, *Detection, Estimation, and Modulation Theory: Part I*. John Wiley & Sons, 1968.

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

[7] L. Gargano and A. A. Rescigno, "Optimally fast data gathering in sensor networks," *Lecture Notes in Computer Science*, vol. 4162, 2006.

[8] C. Florens, M. Franceschetti, and R. McEliece, "Lower bounds on data collection time in sensory networks," *IEEE JSAC*, vol. 22, no. 6, 2004.

[9] B. Krishnamachari, D. Estrin, and S. B. Wicker, "The impact of data aggregation in wireless sensor networks," in *ICDCSW '02*, 2002.

[10] A. Boulis, S. Ganeriwal, and M. B. Srivastava, "Aggregation in sensor networks: An energy-accuracy trade-off," in *1st IEEE Int'l. Wksp. on Sensor Network Protocols and Applications*, 2003.

[11] Y. Yu, B. Krishnamachari, and V. K. Prasanna, "Energy-latency tradeoffs for data gathering in wireless sensor networks," in *IEEE INFOCOM*, 2004.

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

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

[14] ——, "On optimal energy efficient convergecasting in unreliable sensor networks with applications to target tracking," in *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2011.

[15] S. Hariharan, "Communication efficient convergecasting for data fusion in wireless sensor networks," Ph.D. dissertation, The Ohio State University, 2011.

[16] F. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, 1997.

[17] F. C. R. Spieksma, "On the approximability of an interval scheduling problem," *Journal of Scheduling*, vol. 2, no. 5, 1999.

[18] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber, "A unified approach to approximating resource allocation and scheduling," *Journal of the ACM*, vol. 48, no. 5, pp. 735 – 744, 2001.

[19] J. Chuzhoy, R. Ostrovsky, and Y. Rabani, "Approximation algorithms for the job interval selection problem and other related problems," *Mathematics of Operations Research*, vol. 31, 2006.

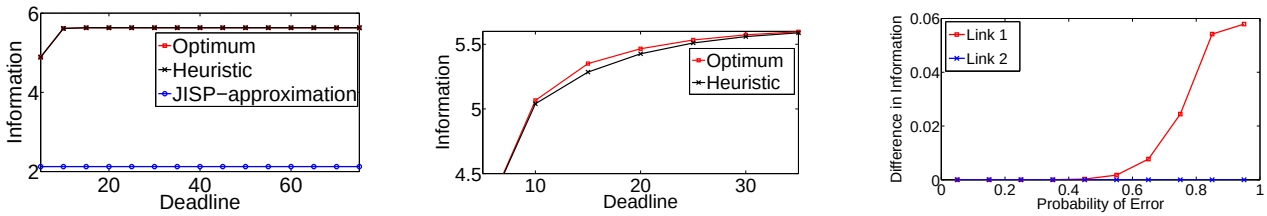
[20] S. K. Sahni, "Algorithms for scheduling independent tasks," *Journal of the ACM*, vol. 23, no. 1, pp. 116–127, 1976.

[21] R. Bar-Yehuda and M. M. Halldórsson, "Scheduling split intervals," *SIAM J. Comput.*, 2006.

[22] D. Joseph, J. Meidanis, and P. Tiwari, "Determining dna sequence similarity using maximum independent set algorithms for interval graphs," *Lecture Notes in Computer Science*, 1992.

Appendix

Proof of Theorem V.2



(a) Link 2 with error probability 0.15 (b) Link 2 with error probability 0.9 (c) Difference in information

Fig. 5. Numerical Results

Proof: We prove this theorem by induction on nodes in the tree from leaves to the root.

At any leaf node l, $X[l,W] = w_l\lambda_l$, since the maximum information that reaches a leaf node is w_l, if l is a source, and 0, otherwise. Therefore, the result holds for leaf nodes.

Assume that the result holds for all nodes at depth h.

Consider any node i at depth $h-1$ from the root of the tree. Suppose that $X[i,W]$ computed by (5) is not maximum. This implies that for at least one child j of i, and for at least one waiting time W', $X[j,W']$ is not maximum. However, this contradicts the induction hypothesis since j is a node that is at depth h from the root of the tree.

Hence, the result follows. ■

Proof of Theorem VI.1

Proof: We show this result by induction on n.

Clearly, for $n=0$, $Z[i,\mathcal{I},s]$ is initialized correctly for each $\mathcal{I} \subseteq \mathcal{J}$, and $s \in \{0,...,W\}$. It now suffices to show that at the end of the i-th iteration, $Z[i,\mathcal{I},s]$ is assigned the correct value for each $\mathcal{I} \subseteq \mathcal{J}$, and $s \in \{0,...,W\}$, assuming the correctness of the $(i-1)$-th iteration. Let $T_i = (v_i, s_i, l_i, u_i)$ denote the i-th time window. If $v_i \notin \mathcal{I}$, then it is clear that $Z[i,\mathcal{I},s] = Z[i-1,\mathcal{I},s]$ as shown in Step 6. Suppose that $v_i \in \mathcal{I}$. Then $Z[i,\mathcal{I},s]$ is maximized by including either a time window T_j that belongs to job v_i with $j<i$, or T_i itself. In the first case, we again have $Z[i,\mathcal{I},s] = Z[i-1,\mathcal{I},s]$ as shown in Steps 6 and 8. In the second case, we must have $Z[i,\mathcal{I},s] = Z[i-1,\mathcal{I}\backslash\{v_i\},s+l_i]+u_i$ by the optimality of the $(i-1)$-th iteration, and the non-increasing ordering of release times. Furthermore, this can only happen when $s+l_i \le W$, and $s \ge s_i$, again by the non-increasing ordering of release times.

Therefore, the result holds for the i^{th} iteration, and hence the proof follows by induction. ■

Proof of Theorem VIII.2

Proof: We prove this result by induction. Consider child k. For any $e \le r$, $m[k,e,w,t] = X[k,w,\lfloor\frac{E_k - eE_T}{E_R}\rfloor f(t,e)$ is the maximum information that reaches node i using child k alone. Hence, the result is true for child k.

Assume that $m[j,e,w,t]$ represents the maximum information that reaches node i using children $j, j+1, ..., k$, when the child j has a waiting time w and transmits for t slots, and node i expends at most e units of energy on receptions.

Consider node $j-1$. Suppose that $m[j-1,e,w,t]$ computed by (11) is not the maximum information that reaches node i. This implies that there exists w', t', and e' satisfying the constraints in (11) such that $m[j,e-e',w',t']$ is not maximum. However, this contradicts the induction hypothesis.

Hence, the result holds for all nodes. ■

Srikanth Hariharan received his Ph.D. in Electrical and Computer Engineering from the Ohio State University in 2011, M.S. in Electrical and Computer Engineering from Purdue University in 2007, and B.Tech. in Electrical Engineering from the Indian Institute of Technology Madras in 2006. He is currently with the Analysis and Optimization department in AT&T Labs. His research interests include analysis and optimization of LTE/UMTS networks, data aggregation in wireless sensor networks, multi-hop wireless scheduling, and wireless security.

Zizhan Zheng (S'07/M'10) received his Ph.D. in Computer Science from The Ohio State University in 2010, M.S. in Computer Science from Peking University, China in 2005, and B.E. in Polymer Engineering from Sichuan University, China in 2002.

He is currently a postdoctoral researcher in Department of Electrical and Computer Engineering at The Ohio State University. His research is in the general area of wireless and sensor networks with focus on network design and optimization.

Ness B. Shroff (F'07) received his Ph.D. degree from Columbia University, NY, in 1994 and joined Purdue University as an Assistant Professor. At Purdue, he became Professor of the School of Electrical and Computer Engineering in 2003 and director of CWSA in 2004, a university-wide center on wireless systems and applications. In July 2007, he joined The Ohio State University as the Ohio Eminent Scholar of Networking and Communications, a chaired Professor of ECE and CSE. He is also a guest chaired professor of Wireless Communications and Networking in the department of Electronic Engineering at Tsinghua University. His research interests span the areas of wireless and wireline communication networks. He is especially interested in fundamental problems in the design, performance, pricing, and security of these networks.

Dr. Shroff is a past editor for IEEE/ACM Trans. on Networking and the IEEE Communications Letters and current editor of the Computer Networks Journal. He has served as the technical program co-chair and general co-chair of several major conferences and workshops, such as the IEEE INFOCOM 2003, ACM Mobihoc 2008, IEEE CCW 1999, and WICON 2008. He was also a co-organizer of the NSF workshop on Fundamental Research in Networking (2003) and the NSF Workshop on the Future of Wireless Networks (2009). Dr. Shroff is a fellow of the IEEE. He received the IEEE INFOCOM 2008 best paper award, the IEEE INFOCOM 2006 best paper award, the IEEE IWQoS 2006 best student paper award, the 2005 best paper of the year award for the Journal of Communications and Networking, the 2003 best paper of the year award for Computer Networks, and the NSF CAREER award in 1996 (his INFOCOM 2005 paper was also selected as one of two runner-up papers for the best paper award).