Towards Optimal Rate Allocation for Data Aggregation in Wireless Sensor Networks[*]

Lu Su, Yan Gao, and Yong Yang
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL, 61801, USA
{lusu2, yangao3, yang25}@illinois.edu

Guohong Cao
Department of Computer Science & Engineering
The Pennsylvania State University
University Park, PA, 16802, USA
gcao@cse.psu.edu

ABSTRACT

This paper aims at achieving optimal rate allocation for data aggregation in wireless sensor networks. We first formulate this rate allocation problem as a network utility maximization problem. Due to its non-convexity, we take a couple of variable substitutions on the original problem and transform it into an approximate problem, which is convex. We then apply duality theory to decompose this approximate problem into a rate control subproblem and a scheduling subproblem. Based on this decomposition, a distributed algorithm for joint rate control and scheduling is designed, and proved to approach arbitrarily close to the optimum of the approximate problem. Finally, we show that our approximate solution can achieve near-optimal performance through both theoretical analysis and simulations.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Network**]: Network Architecture and Design—*Wireless communication*; G.1.6 [**Mathematics of Computing**]: Optimization—*Convex programming*

General Terms

Algorithms, Theory

Keywords

Wireless Sensor Networks, Data Aggregation, Rate Allocation, Scheduling, Network Utility Optimization, Cross Layer Design

1. INTRODUCTION

Data aggregation has been put forward as an essential paradigm for routing in wireless sensor networks [1]. The idea is to use a function like average, max or min to combine the data coming from different sources enroute to eliminate transmission redundancy and thus save energy as well as bandwidth. In recent years, a large spectrum of studies have been conducted on various problems of data aggregation in sensor networks. However, the following fundamental question has not been answered thus far: "*Does there exist an optimal rate allocation for data aggregation which maximizes the utilization of network resources and meanwhile maintains certain fairness among all the sources?*". This paper gives the answer to this question.

Finding the optimal rate allocation for data aggregation in sensor networks can be regarded as a utility-based resource allocation problem. In particular, each source is associated with a utility, which is defined as a function of the source's sending rate. The function value can be conceptually regarded as the quality of information provided by the source. For a given aggregation tree, there exists a unique "maximum utility" rate allocation, at which the network resource utilization is optimal. Meanwhile, certain fairness such as max-min fairness and proportional fairness can be achieved when we choose appropriate utility functions.

The problem of maximizing the network utilities has been explored in the context of both wired [2, 3, 4, 5, 6] and wireless [7, 8, 9, 10] networks, for rate control in unicast or multicast. Although using a similar approach, we show that rate allocation for data aggregation in wireless sensor networks faces unique challenges both theoretically and practically, making this problem a completely different one to which none of the existing solutions can be applied.

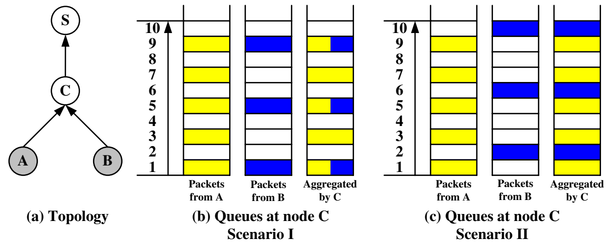


Figure 1: An example of data aggregation constraint

Challenge I: Theoretically, rate allocation for data aggregation is not only subject to the network capacity constraint, but also the *data aggregation constraint* on the *aggregation nodes* (i.e., non-leaf nodes) of the aggregation tree.

Figure 1 provides an intuitive example of the data aggregation constraint. A simple aggregation tree is shown in Fig. 1(a). In this case, two source nodes A and B collect and transmit data to node C, who aggregates the received data and forwards them to the sink S. Fig. 1(b) and (c) illustrate two different scenarios of data aggregation. In either scenario, three columns are displayed, which correspond to three queues maintained by node C. The first two queues store the packets coming from A and B, while the last one (queue C) keeps the packets generated by aggregating A and B's packets.

[*]This work was supported in part by National Science Foundation under grant CNS-0916171.

The packets in each of the three queues are sorted by the timestamps recorded in their headers. In Fig. 1(b) and (c), the vertical axis denotes the timestamp, which indicates the time when the carried data packet is collected. In this paper, we assume that *only the data collected at the same time can be aggregated*. This assumption is valid in many applications of sensor networks, such as target localization and fire alarming, since the data collected at the same time usually contain the information about the same event. For this reason, an aggregated packet has the same timestamp as the raw packets involved in its aggregation. Sometimes, a packet coming from a source node has no coincident packets with the same timestamp from other source nodes, such as the packets with timestamp 3 and 7 in queue A. In this case, the aggregation node does nothing but simply forwards it upwards.

In the first scenario shown in Fig. 1(b), the number of packets stored in queue C is the same as the number of packets in queue A, since the time slots when node B collects data are the subset of the time slots when A collects data. Therefore, to keep the network stable, in other words, to prevent the queue of node C from overflow, the transmission rate of node C should be no less than A's rate. However, this doesn't hold in the second scenario displayed in Fig. 1(c), where the only difference from scenario I is that the timestamps of all the packets in queue B are increased by one which implies that node B postpones all its data collections by one time slot. Surprisingly, this causes a fundamental change in queue C. In particular, no aggregation can be made, since there is no coincident packets of A and B. As a result, the number of packets in queue C is the summation of queue A and B's packets. Therefore, in this scenario, the requirement of stability becomes that C should send faster than the aggregate rate of A and B.

This example reveals the fact that the transmission rate of an aggregation node is constrained by not only the rates of its children but also their *packet timestamp patterns*. The packet timestamp pattern of a node includes two components: the intervals between the timestamps of consecutive packets and the time-offsets of the packets among the nodes who share the same parent.

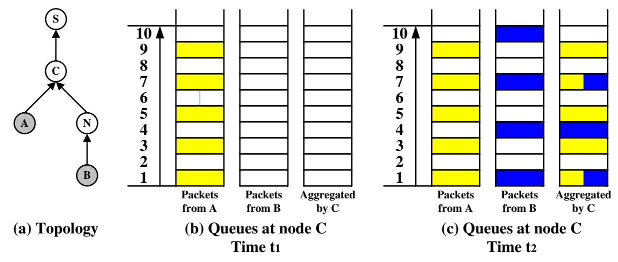


Figure 2: An example of data availability constraint

Challenge II: Practically, rate allocation for data aggregation has an implicit constraint, which is referred to as the *data availability constraint* in this paper. Figure 2 gives us an illustrative example of this constraint. Similar to the previous example, node A and B work as the source nodes. However, B is not directly connected to the aggregation node C. An intermediate node N relays data for B. Suppose at time t_1, as shown in Fig. 2(b), A has delivered some data to C, whereas B's data has not arrived at C since they are delayed at node N. At this moment, although lots of A's packets are waiting in its buffer, node C needs to wait until B's data arrives at time t_2 (for the sake of simplicity, suppose A transmits no data during this period), and then fulfills its aggregation task, as shown in Fig. 2(c). This is because if C chooses to deliver A's packets at time t_1, it has to do the same job again when B's packets arrive. As a result, double amount of traffic is injected into the downstream links by C. This is definitely not an economic solution from the perspective of network utility maximization.

The main purpose of our work is to address the above challenges. We first formulate this rate allocation problem as a network utility maximization problem. Due to its non-convexity, we take a couple of variable substitutions on the original problem and transform it into an approximate problem, which is convex. We then apply duality theory to decompose this approximate problem vertically into a rate control subproblem and a scheduling subproblem that interact through shadow prices. Based on this decomposition, a distributed subgradient algorithm for joint rate control and scheduling is designed, and proved to approach arbitrarily close to the optimum of the approximate problem. Finally, we show that our approximate solution can achieve near-optimal performance through both theoretical analysis and simulations. To the best of our knowledge, this work is the first one to present a joint design of rate allocation and scheduling in the context of data aggregation in wireless sensor networks.

The rest of the paper is organized as follows. Section 2 summarizes the related work. We introduce the system model in Section 3 and formulate the problem in Section 4. In Section 5, the original problem is transformed into a convex approximate problem, with the solution given in Section 6. In Section 7, we explain how the proposed solution is implemented in a decentralized manner. Then, we discuss some related issues in Section 8, and evaluate the performance of the proposed schemes in Section 9. Section 10 concludes the paper.

2. RELATED WORK

In this section, we provide brief summaries of the existing work on sensory data aggregation and network utility maximization respectively, and clarify the novelty of this paper.

Sensory data aggregation becomes a research hotspot after the presentation of the seminal work [1]. A large variety of problems regarding this topic have been extensively studied. Representative problems include: how to construct the most energy efficient aggregation tree [11, 12], how to schedule the transmissions of sensor nodes such that the aggregation delay can be minimized [13, 14], how to maximize or bound the lifetime of the aggregation tree [15, 16], how to secure data aggregation [17, 18], how to derive theoretic bound of aggregation capacity [19, 20], how to achieve fair aggregation [21, 22], etc.

The framework of network utility maximization (NUM) was first developed in the context of wireline network in [2, 3], followed by [4, 5, 6]. The main idea of the framework is based on the decomposition of a system-wide optimization problem. A distributed congestion control mechanism is designed to drive the solutions of the decomposed problems towards the global optimum. Later on, NUM was studied in the context of wireless networks. In wireless networks, this problem is more difficult because of the interference nature of wireless communication. To address this challenge, a scheduling mechanism is integrated into the optimization framework to stabilize the system [7, 8, 9, 10].

This work is the first attempt to utilize NUM framework to explore the optimal rate allocation for sensory data aggregation. The theoretical and practical challenges aforementioned in Section 1 make the problem we target at completely different from previous work, and thus none of the existing solutions can be applied.

3. SYSTEM MODEL

In this section, we explain in detail the system model.

3.1 Aggregation Model

We consider an aggregation tree T rooted at the sink. We denote the set of tree edges (links) by $\mathcal{L} = L(T)$. The sensor nodes on T can be divided into *source nodes* that collect sensory readings, and *aggregation nodes* that aggregate and relay sensory readings. In the rest of the paper, we assume that the source nodes are only at the leaf nodes of the aggregation tree. However, it is possible that a sensor node plays a dual role as both source node and aggregation node. This problem can be easily addressed through a simple graph transformation. In particular, we replace each source node at a non-leaf node by creating a new leaf node and placing the source node in it, and then connect the new leaf node to the non-leaf node where the source node is originally located by a link with infinite capacity[1].

As in most applications of sensor networks, we assume that all the sensor nodes are equipped with the same sensing and communicating devices, and furthermore, the maximum collecting rate of the sensing device is larger than the maximum transmitting rate of the communicating device. The lifetime of the network is divided into slots with equal duration, and each slot is further divided into subslots. The sensing device of each node may or may not collect data at a subslot. Once it decides to work within a subslot, it always collects data at its maximum speed. The data collected within a subslot is encapsulated into a packet, which is referred to as the *Basic Transmission Unit (BTU)*.

3.2 Probabilistic Rate Model

As discussed in Section 1, the transmission rate of an aggregation node is constrained by not only the rates of its children but also their packet timestamp patterns. To characterize the packet timestamp patterns of sensor nodes, we introduce a *probabilistic rate model*, which is defined and explained as below:

Definition 1. **Probabilistic Rate Model.** At each subslot, the sensing device of a node chooses to collect data or not, based on a probability, which is referred to as the *data collection rate* of this node.

Suppose a time slot is composed of 100 subslots, and the data collection rate of a given sensor node is 0.5. Then, it works at roughly 50 randomly selected subslots and sleeps at the rest of time. As a result, it collects around 50 packets (BTU) within a time slot.

We define the *data transmission rate* of a node as the ratio of the number of packets (BTU) this node delivers within a time slot to the number of subslots in each slot. Data transmission rate is actually a normalized rate. Its relation with the genuine transmission rate, which is equal to the number of bits that a node delivers within a time slot, can be reflected by the formula: $x^N = \frac{x^G}{C_B \times N_{sub}}$, where x^N and x^G denote the normalized and genuine data transmission rate, respectively. In addition, C_B is the size of BTU, and N_{sub} represents the number of subslots in each slot. For example, assume the data transmission rate of a node is 1000bps, C_B is 100bits and N_{sub} is 20. Based on the above formula, the normalized data transmission rate is 0.5.

The probabilistic rate model can be considered as a generalization of a node's packet timestamp pattern. This can be better understood after we mathematically formulate the problem in the next section. Additionally, in the sensing tasks which require periodic data collection, the probabilistic rate model can capture the long-term expectation of time-varying time-offsets of the packets from different nodes. It is extremely difficult to mathematically model the fixed time-offsets. More detailed discussion can be found in Section 8.2.

[1]In practice, we do not place this link in any independent set, and thus it has no impact on the scheduling problem.

4. PROBLEM FORMULATION

4.1 Terminology

Link flow: We define link flow as the single-hop data traffic going through a link (tree edge) of the aggregation tree. Conceptually, it includes *source flow* originating at source nodes and *aggregation flow* coming from aggregation nodes. The set of source flows and the set of aggregation flows are denoted by F^S and F^A. In addition, $\mathcal{F} = F^S \bigcup F^A$ represents the set of all the flows on the aggregation tree. For any flow $f \in \mathcal{F}$, we use $\pi(f)$ and $C(f)$ to denote its parent flow and set of children flows, respectively. Finally, $f(l)$ implies the flow which goes through a link $l \in \mathcal{L}$, and $l(f)$ means the link through which a flow $f \in \mathcal{F}$ passes. We denote the normalized rate of each flow $f \in \mathcal{F}$ by x_f. For a source flow $f \in F^S$, its rate is quantitatively equal to the data collection rate of the corresponding source node. We use X_f to denote the interval in which x_f must lie:

$$X_f := \{[m_f, M_f] \quad \text{if } f \in F^S; \quad [0, M_f] \quad \text{if } f \in F^A\}.$$

where m_f and M_f are the lower and upper bounds of x_f.

Queue: At each aggregation node, the packets coming from each of its children flows f is buffered in a queue, denoted by $Q(f)$. The packets in each queue are sorted by their timestamps, as shown in Fig. 1 and Fig. 2. We use $T_t(f)$ to denote the timestamp of the packet on the top of $Q(f)$ (largest timestamp), and $T_b(f)$ to denote the timestamp of the packet at the bottom of $Q(f)$ (smallest timestamp). In addition, the aggregated packets (available data) are stored in a separate queue, waiting for transmission.

4.2 Constraints

In this part, we elaborate on the constraints that our objective function is subject to.

Network Capacity Constraint: Based on the network topology, a conflict graph [23] can be constructed to capture the contention relations among the links. In the conflict graph, each vertex represents a link, and an edge between two vertices implies the contention between the two corresponding links, i.e., they cannot transmit at the same time. Given a conflict graph, we can identify all its independent sets of vertices. The links in an independent set can transmit simultaneously.

Let $\mathcal{I}$ denote the set of independent sets. We represent an independent set, I_i $(i = 1, 2, ..., |\mathcal{I}|)$, as a $|\mathcal{L}|$-dimensional rate vector, which is r^i. In r^i, the lth entry is $r^i_l := \{c_l \quad \text{if } \ l \in I_i; \quad 0 \quad \text{otherwise}\}$ where c_l denotes the capacity of link $l \in \mathcal{L}$. Here we should note that this capacity is a normalized capacity, which is defined as the ratio of the maximum number of packets (BTU) that can be delivered through a link within a time slot to the number of subslots in each slot. The feasible capacity region Π at the link layer is defined as the convex hull of these rate vectors:

$$\Pi := \{r \mid r = \sum_{i=1}^{|\mathcal{I}|} \alpha_i r^i, \ \alpha_i \geq 0, \ \sum_{i=1}^{|\mathcal{I}|} \alpha_i = 1\}.$$

With above notations, now we can formally define the *network capacity constraint* as follows:

$$x_{f(l)} \leq r_l \qquad \text{for all } \ l \in \mathcal{L}.$$

Namely, the rate of each flow must not exceed the amount of capacity allocated to the link it passes through. Let $r := \{(r_{l_1}, r_{l_2}, ..., r_{l_{|\mathcal{L}|}})|\ l_i \in \mathcal{L}\}$, and it should satisfy $r \in \Pi$.

Data Aggregation Constraint: As in the rate control of unicast and multicast scenarios, it is essential to investigate the relationship between the rate of a parent flow and the rates of its children flows so as to stabilize the network. However, the difficulty of achieving this goal in the context of data aggregation is much larger, since a slight change in the packet timestamp pattern of a node may incur significant change in the resulting aggregated packets, as disclosed in Section 1. To overcome this difficulty, we adopt a probabilistic rate model, which is defined in Section 3. Given the rate of a node, this model can generalize all the possibilities of this node's packet timestamp patterns.

Under the probabilistic rate model, the *data aggregation constraint* can be formulated as follows:

$$1 - \prod_{f_c \in C(f)} (1 - x_{f_c}) \le x_f \qquad \text{for all } f \in F^A.$$

Here we give an interpretation of this constraint which may draw a better understanding. We say a node covers a subslot if there exists a packet transmitted by this node whose timestamp is exactly this subslot. Then, $1 - x_f$ denotes the percentage of the subslots which are not covered by the sending node of f. Later, we use the concepts of a flow and its sending node interchangeably. It follows that $\prod_{f_c \in C(f)}(1 - x_{f_c})$ implies the percentage of the subslots not covered by any of f's children nodes. Intuitively, a parent node needs to cover all the subslots covered by at least one child, which is $1 - \prod_{f_c \in C(f)}(1 - x_{f_c})$. By this intuition, the data aggregation constraint is presented.

Data Availability Constraint: From the example shown in Fig. 2, we learn that an aggregation node cannot take any actions before it makes sure that all the packets collected at the same time have arrived. Given a timestamp, after receiving a packet with this timestamp or larger timestamp from each child node, the aggregation node will know that all the packets with this timestamp have arrived. Here we assume that packets arrive in the order of their timestamps. Then, it performs the aggregation and puts the resulting aggregated packet into the queue where the packets available for transmission are stored. Thus, the timestamps of the available packets are constrained within the time interval from the smallest $T_b(f)$ to the smallest $T_t(f)$ among all the queues. Recall that $T_b(f)$ and $T_t(f)$ denote the timestamps of the packets on the bottom and the top of each queue, respectively.

With a binary indicator variable $b_\tau(f)$ defined as below:

$$b_\tau(f) = \begin{cases} 1 & \text{There is a packet with timestamp } \tau \text{ in } Q(f). \\ 0 & \text{otherwise.} \end{cases}$$

In terms of the number of BTUs, we denote the amount of the available data at the sending node of f by λ_f. λ_f can be calculated as follows:

$$\lambda_f = \sum_{\tau = \min_{f_i \in C(f)} T_b(f_i)}^{\min_{f_j \in C(f)} T_t(f_j)} \Big(\bigvee_{f_k \in C(f)} b_\tau(f_k) \Big) \qquad (1)$$

where $\bigvee$ denotes the bitwise operation "OR".

By this formula, we can easily check that in the example shown in Fig. 2, the amount of the available data at node C (stored in queue C) at time t_1 (Fig. 2(b)) and t_2 (Fig. 2(c)) are 0 and 6, respectively. Note that in the scenario happens at t_2, we do not take into account the packet with timestamp 10 in queue B, since it is still unknown at this moment whether node A also has a packet with timestamp 10, and thus cannot mark this packet to be available.

Furthermore, let a_f be the amount of data which can be transmitted by the sending node of f. Then, the *data availability constraint* can be formally defined as follows:

$$a_f \le \lambda_f \qquad \text{for all } f \in F^A.$$

In other words, for each flow $f \in F^A$, it can not deliver more data than the amount of the available data stored at its sending node. Within a time slot, once the available data are all sent out, the sending node can not do more transmission even if some of the queues for its children nodes are not empty. The data availability constraint minimizes the amount of packets a node could inject into the network. By this constraint, for each timestamp, there is at most one packet with this timestamp arriving at the sink. More importantly, the data availability constraint is actually the prerequisite of the data aggregation constraint, since the data aggregation constraint implicitly assumes all the packets from different sources collected at the same time are merged into a single packet.

4.3 Problem Formulation

With the terminologies and constraints defined above, we can now formulate the problem to be solved. We associate each source flow $f \in F^S$ with a utility function $U_f(x_f) : \mathbf{R}_+ \rightarrow \mathbf{R}_+$. In this paper, we assume U_f is continuously differentiable, increasing, and strictly concave. Our objective is to choose the rate of each flow x_f and the allocated capacity of each link r_l so as to maximize the aggregate utility function. We now formulate the problem of optimal rate allocation for data aggregation in sensor networks as the following constrained nonlinear optimization problem:

$$\begin{aligned} \mathbf{P} : \max \quad & \sum_{f \in F^S} U_f(x_f) && && (2) \\ \text{subject to} \quad & x_{f(l)} \le r_l && \text{for all } l \in \mathcal{L} && (3) \\ & a_f \le \lambda_f && \text{for all } f \in F^A && (4) \\ & 1 - \prod_{f_c \in C(f)} (1 - x_{f_c}) \le x_f && \text{for all } f \in F^A && (5) \\ & x \in X, \quad r \in \Pi \end{aligned}$$

In $\mathbf{P}$, the data availability constraint (4) works as the prerequisite of the data aggregation constraint (5). However, it is actually an implicit constraint that need not be considered when solving this optimization problem, although in practice each aggregation node works following this constraint. We will give more detailed explanation on this point in Section 8.1.

By choosing appropriate utility functions, the optimal rate allocation can achieve different fairness models among the flows [2, 3]. For instance, if we let $U_f(x_f) = w_f \ln(x_f)$ for $f \in F^S$, the weighted proportional fairness can be achieved.

5. APPROXIMATE PROBLEM

5.1 Variable Substitution

Though we formulate the problem of optimal rate allocation, it turns out to be a non-convex program, due to the non-convexity of the data aggregation constraint (5). To address this problem, we reorganize the data aggregation constraint and take a log transform on both sides: $\ln(1 - x_f) \le \sum_{f_c \in C(f)} \ln(1 - x_{f_c})$. Next, we substitute x_f of each flow by $\tilde{x}_f = -\ln(1 - x_f)$, where we call

$\tilde{x}_f$ the *transformed rate* of f. By this variable substitution, the data aggregation constraint becomes: $\sum_{f_c \in C(f)} \tilde{x}_{f_c} \le \tilde{x}_f$. In the rest of this paper, we refer to this constraint as the *transformed aggregation constraint*. In addition, based on the feasible region of x_f ($f \in \mathcal{F}$), we can derive the feasible region of $\tilde{x}_f$ as

$$\widetilde{X}_f := \begin{cases} [-\ln(1-m_f), -\ln(1-M_f)] & f \in F^S \\ [0, -\ln(1-M_f)] & f \in F^A \end{cases}$$

where $-\ln(1-m_f) \ge 0$ and $-\ln(1-M_f) < \infty$.

By the variable substitution described above, we transform the data aggregation constraint into a linear constraint. However, this substitution has a side effect, namely, it turns the network capacity constraint into $1 - \exp(-\tilde{x}_{f(l)}) - r_l \le 0$, another non-convex constraint. To overcome this problem, we reorganize this constraint and take a log transform on both sides, then we have $\tilde{x}_{f(l)} \le -\ln(1-r_l)$. Next, we take another variable substitution on r_l: $\tilde{r}_l = -\ln(1-r_l)$ where we call $\tilde{r}_l$ the *transformed allocated capacity* of link $l \in \mathcal{L}$. By this variable substitution, the non-convex constraint is transformed into $\tilde{x}_{f(l)} - \tilde{r}_l \le 0$. We name this constraint as the *transformed capacity constraint*. Recall that r_l is a normalized capacity allocated to link l, and thus it satisfies $0 \le r_l \le 1$. Since $r_l = 1 - \exp(-\tilde{r}_l)$, it can be derived that $0 \le \tilde{r}_l \le \infty$. By substituting $\tilde{r}_l$ for r_l, the original capacity region Π is transformed into a *transformed capacity region* Π':

$$\Pi' := \{\tilde{r} \mid \tilde{r}_{l_i} = -\ln(1-r_{l_i}),\ i = 1, 2, ..., |\mathcal{L}|,\ r \in \Pi\}$$

However, Π' is not a convex region. Figure. 3 illustrates this region transformation. Particularly, Fig. 3(a) shows an example of two-dimensional capacity region Π, and the transformed capacity region Π' is drawn in Fig. 3(b). As can be seen, Π' (the shaded area) is not a convex region.

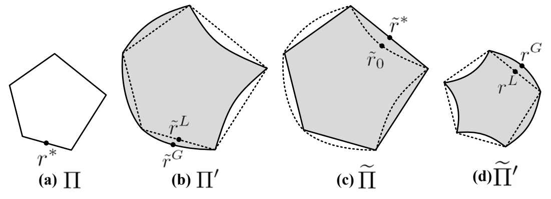


Figure 3: Region transformation

To tackle this problem, we constitute an *approximate transformed capacity region* $\widetilde{\Pi}$, which is convex. Recall that the original capacity region Π is actually the convex hull of the rate vectors of all the independent sets. In fact, these rate vectors are the extreme points of Π, since each of them cannot be represented by the convex combination of others. In the transformed capacity region Π', let $\tilde{r}^i$ denote the point (vector) transformed from the ith extreme point r^i (rate vector of ith independent set) in Π. In $\tilde{r}^i$, the lth entry is $\tilde{r}^i_l := \{\tilde{c}_l \quad \text{if} \quad l \in I_i; \quad 0 \quad \text{otherwise}\}$ where $\tilde{c}_l$ is referred to as the *transformed capacity*, and defined by $\tilde{c}_l = -\ln(1-c_l)$.

Now, we can define the approximate transformed capacity region $\widetilde{\Pi}$ as the convex hull of these transformed rate vectors:

$$\widetilde{\Pi} := \{\tilde{r} \mid \tilde{r} = \sum_{i=1}^{|\mathcal{I}|} \alpha_i \tilde{r}^i,\ \alpha_i \ge 0,\ \sum_{i=1}^{|\mathcal{I}|} \alpha_i = 1\}.$$

It is not difficult to prove that each $\tilde{r}^i$, $i = 1, 2, ..., |\mathcal{I}|$ cannot be represented by the convex combination of others either, and thus is an extreme point of $\widetilde{\Pi}$. Therefore, for each (ith) independent set, there is a one-to-one mapping between its corresponding extreme points in Π (i.e., r^i) and $\widetilde{\Pi}$ (i.e., $\tilde{r}^i$).

Figure 3(c) shows the approximate capacity region $\widetilde{\Pi}$ (the shaded area), which corresponds to the original capacity region Π in Fig. 3(a). As can be seen, despite of the convexity it achieves, it does not cover all the points of the transformed capacity region Π' (the area enclosed by the dashed curve). Furthermore, it includes some points outside the boundary of Π', and this implies that $\widetilde{\Pi}$ may result in some solutions which are not feasible in the original problem. Actually, if we take a reverse variable substitution (i.e., $r_l = 1 - \exp(-\tilde{r}_l)$) on each point $\tilde{r} \in \widetilde{\Pi}$, a new region denoted by $\widetilde{\Pi}'$ is attained, and shown in Fig. 3(d) (the shaded area). As one can see, it does have some points outside the original capacity region Π (the area enclosed by the dashed curve).

However, in our algorithm that will be introduced in the next section, we do not map the solution in $\widetilde{\Pi}$ to Π in this way, namely, through $r_l = 1 - \exp(-\tilde{r}_l)$. Instead, we design a safe mapping scheme, which can guarantee that there always exists a feasible point in Π, which corresponds to the solution attained in the context of $\widetilde{\Pi}$.

Based on the definition of $\widetilde{\Pi}$, any point in $\widetilde{\Pi}$, say $\tilde{r}_0$, can be expressed as $\tilde{r}_0 = \sum_{i=1}^{|\mathcal{I}|} \alpha_i \tilde{r}^i$. Its counterpart in the original problem is $r^G := \{(r^G_l) \mid r^G_l = 1 - \exp(-\tilde{r}_{0l}),\ l \in \mathcal{L}\}$, and it may not be located inside Π. However, in Π, we can always find a point, which is $r^L = \sum_{i=1}^{|\mathcal{I}|} \alpha_i r^i$ where each α_i equals the α_i in $\tilde{r}_0 = \sum_{i=1}^{|\mathcal{I}|} \alpha_i \tilde{r}^i$. By this mapping scheme, wherever the optimal solution is located in $\widetilde{\Pi}$, our algorithm can identify a corresponding feasible solution inside Π. In the rest of this paper, we refer to r^G as the *genuine mapping point* of $\tilde{r}_0$, and r^L as the *linear mapping point* of $\tilde{r}_0$. Similarly, given a point $r_0 = \sum_{i=1}^{|\mathcal{I}|} \alpha_i r^i$ inside Π, we can define $\tilde{r}^G := \{(\tilde{r}^G_l) \mid \tilde{r}^G_l = -\ln(1-r_{0l}),\ l \in \mathcal{L}\}$ and $\tilde{r}^L = \sum_{i=1}^{|\mathcal{I}|} \alpha_i \tilde{r}^i$ as the *genuine mapping point* and the *linear mapping point* of r_0 in $\widetilde{\Pi}$.

Now, we can formally define the *approximate problem* as follows:

$$\widetilde{\mathbf{P}}: \ \max \quad \sum_{f \in F^S} U_f(1 - \exp(-\tilde{x}_f)) \qquad (6)$$

$$\text{subject to} \quad \tilde{x}_{f(l)} - \tilde{r}_l \le 0 \qquad \text{for all } l \in \mathcal{L} \qquad (7)$$

$$\sum_{f_c \in C(f)} \tilde{x}_{f_c} - \tilde{x}_f \le 0 \qquad \text{for all } f \in F^A \qquad (8)$$

$$\tilde{x} \in \widetilde{X}, \quad \tilde{r} \in \widetilde{\Pi}$$

According to [24] (Chapter 3.2.4), since $1 - \exp(-\tilde{x}_f)$ is a strictly concave and increasing function, the objective function (6) remains strictly concave and increasing. Thus, $\widetilde{\mathbf{P}}$ is a convex problem, and always has a unique maximizer. Once we identify this maximizer, we can use its linear mapping point in Π as the *approximate solution* of $\mathbf{P}$.

5.2 Approximation Analysis

In this subsection, we provide some theoretical analysis on both the original problem $\mathbf{P}$ and the approximate problem $\widetilde{\mathbf{P}}$.

Theorem 1. The optimal solution of $\mathbf{P}$ ($\widetilde{\mathbf{P}}$) must be attained on the boundary of Π ($\widetilde{\Pi}$).

PROOF. Here we only show the proof for $\mathbf{P}$, since the proof for $\widetilde{\mathbf{P}}$ is similar. By contradiction, suppose the optimal solution of $\mathbf{P}$, denoted by r^*, is a strictly interior point of Π^2. Since in Π, the components of r^* only appear in the network capacity constraint (i.e., $x_{f(l)} \le r_l$), we do not need to check other constraints. At optimality, the network capacity constraint may or may not be active (we say a constraint is active if it attains equality). If it is not active, $x_{f(l)}$ will not change if we increase r_l. On the other hand, if it is active, $x_{f(l)}$ will go up to some extent with the increase of r_l. As a result, the objective value will be improved, since it's strictly increasing with $x_{f(l)}$. Since r^* is an interior point, there must exist some room to increase some components of r^*, without changing the others. This conflicts the assumption that r^* is the optimal solution. Therefore, r^* must be located on the boundary of Π. □

In Π ($\widetilde{\Pi}$), which is a compact $|\mathcal{L}|$-dimensional polyhedron, each facet of its boundary is defined by the convex hull of $|\mathcal{L}|$ extreme points. Thus, the optimal solution of $\mathbf{P}$ ($\widetilde{\mathbf{P}}$) can be expressed as $r^* = \sum_{i=1}^{|\mathcal{L}|} \alpha_i r^i$ ($\tilde{r}^* = \sum_{i=1}^{|\mathcal{L}|} \alpha_i \tilde{r}^i$). In addition, the approximate solution, which is the linear mapping point of $\tilde{r}^*$, is also located at the boundary of Π.

Now, we are interested in how far our approximate solution is from the optimal solution. In other words, we want to know the difference between our approximate objective value and the optimal objective value. We first introduce some notations. For any point r_0 ($\tilde{r}_0$) in Π ($\widetilde{\Pi}$), we define $\mathbf{P}(r_0)$ ($\widetilde{\mathbf{P}}(\tilde{r}_0)$) as the optimization problem $\mathbf{P}$ ($\widetilde{\mathbf{P}}$) when r ($\tilde{r}$) is fixed to be r_0 ($\tilde{r}_0$). In addition, let $\mathbf{P}^*(r_0)$ ($\widetilde{\mathbf{P}}^*(\tilde{r}_0)$) be the optimal objective value of $\mathbf{P}(r_0)$ ($\widetilde{\mathbf{P}}(\tilde{r}_0)$). Suppose r^* ($\tilde{r}^*$) is the global optimal solution in Π ($\widetilde{\Pi}$), we use $\mathbf{P}^* = \mathbf{P}^*(r^*)$ ($\widetilde{\mathbf{P}}^* = \widetilde{\mathbf{P}}^*(\tilde{r}^*)$) to denote the global optimal objective value of $\mathbf{P}$ ($\widetilde{\mathbf{P}}$).

Then, we investigate the performance of the approximate solution. Suppose the objective value of our approximate solution is $\hat{\mathbf{P}}^*$. In the rest of this section, we first show that the difference between the global optimal objective value of $\mathbf{P}$ (i.e., $\mathbf{P}^*$) and $\hat{\mathbf{P}}^*$ is bounded by $\widetilde{\mathbf{P}}^* - \hat{\mathbf{P}}^*$ through Theorem 2, and then give a looser but simpler bound by Theorem 3.

Theorem 2. The optimal objective value of the original problem $\mathbf{P}$ is upper bounded by the optimal objective value of the approximate problem $\widetilde{\mathbf{P}}$.

PROOF. Let the point in Π which maximizes $\mathbf{P}$ be r^*, as shown in Fig. 3(a). Thus, r^* can be expressed as $r^* = \sum_{i=1}^{|\mathcal{L}|} \alpha_i r^i$. Suppose its genuine mapping point in $\widetilde{\Pi}$ is $\tilde{r}^G$. As can be seen in Fig. 3(b), it may not be inside $\widetilde{\Pi}$. However, in $\widetilde{\Pi}$, we can always find the linear mapping point of r^*, which is denoted by $\tilde{r}^L$ and shown in Fig. 3(b). Since the function $f(x) = -\ln(1-x)$ is strictly convex, it can be derived that for each $l \in \mathcal{L}$, $\tilde{r}_l^G = -\ln(1 - r_l^*) \le \sum_{i=1}^{|\mathcal{L}|} \alpha_i(-\ln(1 - r_l^i)) = \sum_{i=1}^{|\mathcal{L}|} \alpha_i \tilde{r}_l^i = \tilde{r}_l^L$. Similar to the proof of Theorem 1, we can show that $\widetilde{\mathbf{P}}^*(\tilde{r}^G) \le \widetilde{\mathbf{P}}^*(\tilde{r}^L)$ by moving each component of $\tilde{r}^G$ towards $\tilde{r}^L$. Since $\mathbf{P}^* = \widetilde{\mathbf{P}}^*(\tilde{r}^G)$ and $\widetilde{\mathbf{P}}^*(\tilde{r}^L) \le \widetilde{\mathbf{P}}^*$, it can be concluded that $\mathbf{P}^* \le \widetilde{\mathbf{P}}^*$. □

By Theorem 2, the approximation ratio of our solution can be bounded by $\frac{\widetilde{\mathbf{P}}^* - \hat{\mathbf{P}}^*}{\hat{\mathbf{P}}^*}$. Next, we give a looser but simpler bound of $\mathbf{P}^* - \hat{\mathbf{P}}^*$.

Theorem 3. Suppose that the optimal solution of $\widetilde{\mathbf{P}}$ is $\tilde{r}^*$, and its linear mapping point in Π, i.e., the approximate solution is r^L. Furthermore, let $\tilde{r}_0$ be r^L's corresponding genuine mapping point in $\widetilde{\Pi}$. Then, the value of $\mathbf{P}^* - \hat{\mathbf{P}}^*$ is bounded by $\mu^{\alpha * T}(\tilde{r}^* - \tilde{r}_0)$, where $\mu^{\alpha *}$ represents the vector of the optimal dual variables of problem $\widetilde{\mathbf{P}}(\tilde{r}_0)$.

PROOF. As can be seen, $\tilde{r}^*$ and $\tilde{r}_0$ are shown in Fig. 3(c), and r^L is shown in Fig. 3(d). Since r^L is the approximate solution, by Theorem 2, $\mathbf{P}^* - \mathbf{P}^*(r^L)$ is bounded by $\widetilde{\mathbf{P}}^*(\tilde{r}^*) - \mathbf{P}^*(r^L)$, which is further equal to $\widetilde{\mathbf{P}}^*(\tilde{r}^*) - \widetilde{\mathbf{P}}^*(\tilde{r}_0)$. Based on the theory of perturbation and sensitivity (Chapter 5.6 in [24]), we denote the perturbed version of the optimization problem $\widetilde{\mathbf{P}}(\tilde{r}_0)$ by $\widetilde{\mathbf{P}}_{\tilde{\mathbf{r}}_0}$, in which the transformed capacity constraint is replaced by $\tilde{x}_{f(l)} - \tilde{r}_{0l} \le u_l$. Here $u := (u_l, l \in \mathcal{L})$ is the vector of perturbation variables. It is evident that $\widetilde{\mathbf{P}}_{\tilde{\mathbf{r}}_0}$ coincides with problem $\widetilde{\mathbf{P}}(\tilde{r}_0)$ when u is a zero vector. On the other hand, when u_l is positive it means that we have relaxed the transformed capacity constraint of link l.

We denote the optimal objective value of $\widetilde{\mathbf{P}}_{\tilde{\mathbf{r}}_0}$ at u by $\widetilde{\mathbf{P}}^*_{\tilde{\mathbf{r}}_0}(u)$. According to [24] (Chapter 5.6.1), since problem $\widetilde{\mathbf{P}}(\tilde{r}_0)$ is concave, $\widetilde{\mathbf{P}}^*_{\tilde{\mathbf{r}}_0}(u)$ is a concave function of u. It follows that $\widetilde{\mathbf{P}}^*_{\tilde{\mathbf{r}}_0}(u) \le \widetilde{\mathbf{P}}^*_{\tilde{\mathbf{r}}_0}(0) + \mu^{\alpha * T} u$. Therefore, let $u = \tilde{r}^* - \tilde{r}_0$, the difference between $\mathbf{P}^*$ and $\mathbf{P}^*(r^L)$ can be bounded as follows: $\mathbf{P}^* - \mathbf{P}^*(r^L) \le \widetilde{\mathbf{P}}^*(\tilde{r}^*) - \widetilde{\mathbf{P}}^*(\tilde{r}_0) = \widetilde{\mathbf{P}}^*_{\tilde{\mathbf{r}}_0}(u) - \widetilde{\mathbf{P}}^*_{\tilde{\mathbf{r}}_0}(0) \le \mu^{\alpha * T} u = \mu^{\alpha * T}(\tilde{r}^* - \tilde{r}_0)$.

Let $f(x) = -\ln(1-x)$, and thus $f^{-1}(y) = 1 - \exp(-y)$. As previously discussed, $\tilde{r}^* = \sum_{i=1}^{|\mathcal{L}|} \alpha_i \tilde{r}^i$. It follows that $r^G = f^{-1}(\sum_{i=1}^{|\mathcal{L}|} \alpha_i \tilde{r}^i)$ (for the sake of simplicity, here we use the notation of a vector to delegate all of its components.) and $r^L = \sum_{i=1}^{|\mathcal{L}|} \alpha_i r^i$. Similar to the proof of Theorem 2, it can be proved that $r^G \ge r^L$ (i.e., $r_l^G \ge r_l^L$). Since $f(x)$ is strict increasing, and $\tilde{r}_0 = f(r^L)$, it can be inferred that $\tilde{r}^* \ge \tilde{r}_0$ (i.e., $\tilde{r}_l^* \ge \tilde{r}_{0l}$). Therefore, each component of $\tilde{r}^* - \tilde{r}_0$ is nonnegative. Furthermore, since $\tilde{r}_0 = f(\sum_{i=1}^{|\mathcal{L}|} \alpha_i f^{-1}(\tilde{r}^i))$, $\mu^{\alpha * T}(\tilde{r}^* - \tilde{r}_0)$ is a function of $\mu^{\alpha *}$ and $\tilde{r}^i$, $i = 1, 2, ..., |\mathcal{L}|$. □

From this bound, it can be seen that $\mathbf{P}^* - \hat{\mathbf{P}}^*$ is proportional to the difference between $\tilde{r}^*$ and $\tilde{r}_0$. Actually, it is not difficult to show that when the capacity of each link decreases, the difference between $\tilde{r}^*$ and $\tilde{r}_0$ will drop accordingly. However, this does not necessarily means that $\mu^{\alpha * T}(\tilde{r}^* - \tilde{r}_0)$ will also drop, since $\mu^{\alpha *}$ may increase with the decrease of capacities[3]. In fact, $\mu^{\alpha *}$ depends on the particular utility function we choose, and thus there is no universal conclusion on this point. In Section 9, we will show an example in which $\mathbf{P}^* - \hat{\mathbf{P}}^*$ drops when the capacity of each link is reduced.

6. CROSS LAYER DESIGN VIA DUAL DECOMPOSITION

6.1 The Dual Problem

Solving $\widetilde{\mathbf{P}}$ directly requires global coordination of all flows, which is impractical in a distributed environment such as sensor networks. Since $\widetilde{\mathbf{P}}$ is a convex program with compact feasible region, strong duality can be achieved[4] (Chapter 5.2.3 in [24]). Therefore, there exists a unique maximizer $(\tilde{x}^*, \tilde{r}^*)$ for $\widetilde{\mathbf{P}}$, which can be attained by a distributed algorithm derived via formulating

[2]In fact, a solution also includes the rate x, here we only consider the capacity r simply for the ease of expression.

[3]For more detailed explanation on $\mu^{\alpha *}$, please refer to Section 6.

[4]Slater's condition can be guaranteed by assuming there exist vectors $\tilde{x} \in \widetilde{X}$ and $\tilde{r} \in \widetilde{\Pi}$ which satisfy all the constraints, i.e., strictly feasible points exist.

and solving the Lagrange dual problem of $\widetilde{\mathbf{P}}$. In order to achieve this, we first take a look at the Lagrangian of $\widetilde{\mathbf{P}}$:

$$L(\tilde{x},\tilde{r},\mu^\alpha,\mu^\beta) = \sum_{f\in F^S} U_f(1-\exp(-\tilde{x}_f)) - \sum_{l\in\mathcal{L}} \mu_l^\alpha(\tilde{x}_{f(l)} - \tilde{r}_l) - \sum_{f\in F^A} \mu_f^\beta(\sum_{f_c\in C(f)} \tilde{x}_{f_c} - \tilde{x}_f) .$$

In $L(\tilde{x},\tilde{r},\mu^\alpha,\mu^\beta)$, $\mu^\alpha := (\mu_l^\alpha, l\in\mathcal{L})$ and $\mu^\beta := (\mu_f^\beta, f\in\mathcal{F})$ are vectors of Lagrangian multipliers, corresponding to the transformed capacity constraint (7) and the transformed aggregation constraint (8), respectively. They are also interpreted as the "shadow prices" of the constraints, which can be understood as the "costs" a flow will be charged if it violates the constraints.

Since it can be derived that

$$\sum_{f\in F^A} \mu_f^\beta(\sum_{f_c\in C(f)} \tilde{x}_{f_c} - \tilde{x}_f) = \sum_{f\in F^A} \mu_f^\beta \sum_{f_c\in C(f)} \tilde{x}_{f_c} - \sum_{f\in F^A} \mu_f^\beta \tilde{x}_f$$
$$= \sum_{f\in\mathcal{F}} \mu_{\pi(f)}^\beta \tilde{x}_f - \sum_{f\in F^A} \mu_f^\beta\tilde{x}_f = \sum_{f\in F^S} \mu_{\pi(f)}^\beta \tilde{x}_f + \sum_{f\in F^A} (\mu_{\pi(f)}^\beta - \mu_f^\beta)\tilde{x}_f$$

and $\sum_{l\in\mathcal{L}} \mu_l^\alpha \tilde{x}_{f(l)} = \sum_{f\in\mathcal{F}} \mu_{l(f)}^\alpha \tilde{x}_f$, we reorganize the Lagrangian as follows:

$$L(\tilde{x},\tilde{r},\mu^\alpha,\mu^\beta) = \sum_{f\in F^S} \Big[U_f(1-\exp(-\tilde{x}_f)) - (\mu_{l(f)}^\alpha + \mu_{\pi(f)}^\beta)\tilde{x}_f\Big] + \sum_{f\in F^A} \Big[(-\mu_{l(f)}^\alpha - \mu_{\pi(f)}^\beta + \mu_f^\beta)\tilde{x}_f\Big] + \sum_{l\in\mathcal{L}} \mu_l^\alpha \tilde{r}_l.$$

The dual of the primal problem $\widetilde{\mathbf{P}}$ is:

$$\widetilde{\mathbf{D}}: \min_{\mu^\alpha,\mu^\beta\geq 0} \quad D(\mu^\alpha,\mu^\beta),$$

where the dual objective function $D(\mu^\alpha,\mu^\beta)$ is given as

$$D(\mu^\alpha,\mu^\beta) := \max_{\tilde{x}\in\widetilde{X},\ \tilde{r}\in\widetilde{\Pi}} \quad L(\tilde{x},\tilde{r},\mu^\alpha,\mu^\beta)$$

In the dual objective function, the Lagrangian multipliers (shadow prices) μ^α and μ^β, serve as the dual variables. Furthermore, $D(\mu^\alpha,\mu^\beta)$ can be decomposed into two separate optimization problems: $D(\mu^\alpha,\mu^\beta) = D_1(\mu^\alpha,\mu^\beta) + D_2(\mu^\alpha)$. $D_1(\mu^\alpha,\mu^\beta)$ and $D_2(\mu^\alpha)$ are defined below:

$$D_1(\mu^\alpha,\mu^\beta) := \max_{\tilde{x}\in\widetilde{X}} \quad \sum_{f\in F^S} \Big[U_f(1-\exp(-\tilde{x}_f)) - (\mu_{l(f)}^\alpha + \mu_{\pi(f)}^\beta)\tilde{x}_f\Big] + \sum_{f\in F^A} \Big[(-\mu_{l(f)}^\alpha - \mu_{\pi(f)}^\beta + \mu_f^\beta)\tilde{x}_f\Big]$$
$$D_2(\mu^\alpha) := \max_{\tilde{r}\in\widetilde{\Pi}} \quad \sum_{l\in\mathcal{L}} \mu_l^\alpha \tilde{r}_l$$

Among them, $D_1(\mu^\alpha,\mu^\beta)$ denotes the *rate allocation problem*, while $D_2(\mu^\alpha)$ is the *scheduling problem*. In particular, the rate allocation problem aims at finding the rate of each source node that maximizes the aggregate utilities of all sources, subject to the constraint that the system is stable under some scheduling policy, while the scheduling problem focuses on finding a scheduling policy that stabilizes the system, for any rate vector of sources picked by the rate allocation problem. In the rest of this section, we will first elaborate on these two problems separately, and then explain how to develop a cross-layer joint design of them.

6.2 Interpretation of the Prices

Before proceeding with the decoupled problems, we first provide detailed explanation on the aforementioned shadow prices μ^α and μ^β. Theoretically, these prices represent the "costs" a flow will be charged if it violates the constraints. In practice, they imply the congestion information that the network elements need to share with each other, so that the traffic rates on different links of the network can be adjusted appropriately.

Let us first take a look at μ^α, which corresponds to the transformed capacity constraint in $\widetilde{\mathbf{P}}$. When a flow f violates this constraint (i.e., $\tilde{x}_{f(l)} > \tilde{r}_l$), if f increases its rate for $d\tilde{x}_{f(l)}$, a cost of $\mu_l^\alpha d\tilde{x}_{f(l)}$ should be charged. Next, we give a practical interpretation of this cost in the context of the original problem $\mathbf{P}$. Since $\tilde{x}_{f(l)} = -\ln(1-x_{f(l)})$, it can be derived that $d\tilde{x}_{f(l)} = \frac{1}{1-x_{f(l)}}dx_{f(l)}$. Therefore, the cost charged with respect to $x_{f(l)}$ is $\mu_l^\alpha d\tilde{x}_{f(l)} = \mu_l^\alpha \frac{1}{1-x_{f(l)}}dx_{f(l)}$.

In this paper, we call $\mu_l^\alpha \frac{1}{1-\hat{x}_{f(l)}}$ the *link price* of flow f when it passes data at a rate of $x_{f(l)} = \hat{x}_{f(l)}$. With link price, when a flow f violates the network capacity constraint (3) in the original problem $\mathbf{P}$, i.e., $x_{f(l)} > r_l$, the total cost it needs to pay can be calculated as follows: $\int_{r_l}^{x_{f(l)}} \mu_l^\alpha \frac{1}{1-\hat{x}_{f(l)}} d\hat{x}_{f(l)} = \mu_l^\alpha(\tilde{x}_{f(l)} - \tilde{r}_l)$. As can be seen, it is quantitatively equal to the cost calculated in the context of $\widetilde{\mathbf{P}}$, which is $\mu_l^\alpha(\tilde{x}_{f(l)} - \tilde{r}_l)$.

On the other hand, μ^β corresponds to the transformed aggregation constraint in $\widetilde{\mathbf{P}}$. When the aggregate transformed rates of f's children flows are larger than f's transformed rate (i.e., $\sum_{f_c\in C(f)} \tilde{x}_{f_c} > \tilde{x}_f$), the total cost paid by them to f for its efforts of aggregating packets is $\mu_f^\beta(\sum_{f_c\in C(f)} \tilde{x}_{f_c} - \tilde{x}_f)$.

When at optimality, according to the Karush-Kuhn-Tucker conditions, only the prices corresponding to active constraints are positive, which implies the price of an uncongested link is zero.

6.3 The Rate Allocation Problem

The rate allocation problem can be further divided as follows:

$$D_1(\mu^\alpha,\mu^\beta) = \sum_{f\in F^S} \max_{\tilde{x}_f\in\widetilde{X}_f} \Phi(\tilde{x}_f) + \sum_{f\in F^A} \max_{\tilde{x}_f\in\widetilde{X}_f} \Psi(\tilde{x}_f)$$

$$\text{where} \quad \Phi(\tilde{x}_f) = U_f(1-\exp(-\tilde{x}_f)) - (\mu_{l(f)}^\alpha + \mu_{\pi(f)}^\beta)\tilde{x}_f$$
$$\Psi(\tilde{x}_f) = (-\mu_{l(f)}^\alpha - \mu_{\pi(f)}^\beta + \mu_f^\beta)\tilde{x}_f.$$

In other words, the rate allocation problem can be solved through separately solving the optimization problem of each source flow (i.e., $\max_{\tilde{x}_f\in\widetilde{X}_f} \Phi(\tilde{x}_f)$), and each aggregation flow (i.e., $\max_{\tilde{x}_f\in\widetilde{X}_f} \Psi(\tilde{x}_f)$). Recall that μ^α and μ^β are the costs a flow will be charged if it violates the constraints. $\Phi(\tilde{x}_f)$ and $\Psi(\tilde{x}_f)$ actually represent the "net benefit" of a flow.

Let us first study the optimization problem of each source flow $f\in F^S$. As previously discussed, $U_f(1-\exp(-\tilde{x}_f))$ is strictly concave and twice continuously differentiable. Consequently, $\Phi(\tilde{x}_f)$ is strictly concave and smooth, and thus has a unique maximizer when $\frac{d\Phi(\tilde{x}_f)}{d\tilde{x}_f} = 0$. Thus, given a valid utility function, the optimal solution can be easily identified. For example, assume $U_f(.) = \ln(.)$, it follows that $\frac{d\Phi(\tilde{x}_f)}{d\tilde{x}_f} = \frac{\exp(-\tilde{x}_f)}{1-\exp(-\tilde{x}_f)} - (\mu_{l(f)}^\alpha + \mu_{\pi(f)}^\beta) = 0$ from where the maximizer can be solved as below:

$$\tilde{x}_f^* = -\ln\Big(\frac{\mu_{l(f)}^\alpha + \mu_{\pi(f)}^\beta}{\mu_{l(f)}^\alpha + \mu_{\pi(f)}^\beta + 1}\Big).$$

When taking into account the feasible range of $\tilde{x}_f$, which is $\widetilde{X}_f = [-\ln(1-m_f), -\ln(1-M_f)]$, the optimal value of $\tilde{x}_f$ given μ^α and μ^β should be

$$\tilde{x}_f(\mu^\alpha, \mu^\beta) = \arg\max_{\tilde{x}_f \in \widetilde{X}_f} \Phi(\tilde{x}_f)$$
$$= \begin{cases} \tilde{x}_f^* & \text{if } -\ln(1-m_f) \le \tilde{x}_f^* \le -\ln(1-M_f) \\ -\ln(1-m_f) & \text{if } \tilde{x}_f^* < -\ln(1-m_f) \\ -\ln(1-M_f) & \text{if } \tilde{x}_f^* > -\ln(1-M_f) \end{cases} \quad (9)$$

On the other hand, for each aggregation flow $f \in F^A$, since $\frac{d\Psi(\tilde{x}_f)}{d\tilde{x}_f} = -\mu^\alpha_{l(f)} - \mu^\beta_{\pi(f)} + \mu^\beta_f$ is a constant, it follows that given the feasible range $\widetilde{X}_f = [0, -\ln(1-M_f)]$, together with μ^α and μ^β, the optimal value of $\tilde{x}_f$ can be calculated as below:

$$\tilde{x}_f(\mu^\alpha, \mu^\beta) = \arg\max_{\tilde{x}_f \in \widetilde{X}_f} \Psi(\tilde{x}_f)$$
$$= \begin{cases} 0 & \text{if } \mu^\beta_f < \mu^\alpha_{l(f)} + \mu^\beta_{\pi(f)} \\ -\ln(1-M_f) & \text{if } \mu^\beta_f > \mu^\alpha_{l(f)} + \mu^\beta_{\pi(f)} \\ \text{any value in } \widetilde{X}_f & \text{otherwise} \end{cases} . \quad (10)$$

As previously discussed, strong duality holds in $\widetilde{\mathbf{P}}$, and thus there is no duality gap. Thereby, the optimal dual variables (prices) μ^α and μ^β exist (Proposition 5.1.4 in [25]), denoted as $\mu^{\alpha*}$ and $\mu^{\beta*}$. If $\mu^{\alpha*} > 0$ and $\mu^{\beta*} > 0$ are dual optimal, then $\tilde{x}_f(\mu^{\alpha*}, \mu^{\beta*})$ is also primal optimal, given that $\tilde{x}_f$ is primal feasible (Proposition 5.1.5 in [25]). In other words, once the optimal prices $\mu^{\alpha*}$ and $\mu^{\beta*}$ are available, the optimal rate $\tilde{x}_f^*$ can be achieved. The role of μ^α and μ^β is two-fold. First, they serve as the pricing signal for a flow to adjust its rate. Second, they decouple the primal problem, i.e., the global utility optimization into individual rate optimization of each flow.

6.4 The Scheduling Problem

We now turn to the scheduling problem $D_2(\mu^\alpha)$. It is actually a NP-hard problem, since it is equivalent to the maximum weighted independent set problem over the conflict graph. Actually, the conflict graph depends on the underlying interference model. In this paper, we consider node-exclusive interference model, i.e., links that share a common node cannot transmit or receive simultaneously. This model has been used in many existing works [7, 8, 9] on network utility maximization. With the node exclusive interference model, the scheduling problem can be reduced to the maximum weighted matching problem, which is polynomial-time solvable. However, the existing polynomial-time solution [26] requires centralized implementation. In [27], a simple distributed approximate algorithm is presented, which is at most a factor of 2 away from the maximum, and has a linear running time $O(|\mathcal{L}|)$. We utilize this algorithm to solve the scheduling problem $D_2(\mu^\alpha)$ in a distributed manner.

Actually, the rate control strategy proposed in this paper is a general framework and thus can be extended to other interference models. For any interference model, as long as an appropriate algorithm can be designed to solve the scheduling problem $D_2(\mu^\alpha)$, it can be integrated with our framework.

Additionally, in some applications of sensor networks, the duty-cycle of the sensor nodes further complicate the scheduling problem [28, 29]. We will try to address this challenge in our future work.

6.5 Subgradient Algorithm

Now let us see how we can minimize the dual objective function $D(\mu^\alpha, \mu^\beta)$. Gradient-based methods are, in general, attractive approaches to carry out minimizations of this type. Unfortunately, in our case, $D(\mu^\alpha, \mu^\beta)$ is nondifferentiable, and therefore its gradient may not always exist. This is because in general, differentiability of the dual requires a unique primal optimizer, whereas in our case, the optimal values of $\tilde{x}_f$ $(f \in F^A)$ can be non-unique. Furthermore, $D_2(\mu^\alpha)$ is a piecewise linear function and not differentiable. Therefore, we choose to use subgradient method to solve this problem.

The subgradient algorithm that we propose next is based on the subgradient method developed by N. Z. Shor (Chapter 2 in [30]). In our problem, although the dual gradient does not exist, subgradients do. Based on Proposition 6.1.1 of [25], we adjust μ^α and μ^β in the opposite direction to the subgradients:

$$\mu^\alpha_l(t+1) = \left[\mu^\alpha_l(t) - h(t)\frac{\partial D(\mu^\alpha(t), \mu^\beta(t))}{\partial \mu^\alpha_l}\right]^+ \quad (11)$$
$$= \left[\mu^\alpha_l(t) + h(t)(\tilde{x}_{f(l)}(\mu^\alpha(t), \mu^\beta(t)) - \tilde{r}_l(\mu^\alpha(t)))\right]^+$$

$$\mu^\beta_f(t+1) = \left[\mu^\beta_f(t) - h(t)\frac{\partial D(\mu^\alpha(t), \mu^\beta(t))}{\partial \mu^\beta_f}\right]^+ \quad (12)$$
$$= \left[\mu^\beta_f(t) + h(t)(\sum_{f_c \in C(f)} \tilde{x}_{f_c}(\mu^\alpha(t), \mu^\beta(t)) - \tilde{x}_f(\mu^\alpha(t), \mu^\beta(t)))\right]^+$$

In the above formulas, the $\tilde{x}_f(\mu^\alpha, \mu^\beta)$ and $\tilde{r}_l(\mu^\alpha)$ are the maximizers of $D_1(\mu^\alpha, \mu^\beta)$ and $D_2(\mu^\alpha)$, given μ^α and μ^β; $h(t)$ is a positive scalar stepsize (note that the unit of t is time slot, not subslot); '+' denotes the projection onto the set $\mathbf{R}_+$ of non-negative real numbers.

Equation (11) reflects the law of supply and demand. If the demand of a flow f for bandwidth $\tilde{x}_{f(l)}$ exceeds its supply $\tilde{r}_l$, the transformed capacity constraint is violated. Thus, the price μ^α_l is raised. Otherwise, μ^α_l is reduced. Similarly, in (12), if the children flows $f_c \in C(f)$ demand an aggregate rate higher than the rate of its parent flow f, the transformed aggregation constraint is violated. Thus, the price μ^β_f is raised. Otherwise, μ^β_f is reduced.

6.6 Convergence Analysis

In this subsection, we justify the convergence property of the subgradient algorithm. Subgradient may not be a direction of descent, but makes an angle less than 90 degrees with all descent directions. Using results on the convergence of the subgradient method [25, 30], we show that, for a constant stepsize h, the algorithm is guaranteed to converge to within a neighborhood of the optimal value. The reason why we choose a constant stepsize is that it is convenient for distributed implementation. Since the usual convergence criterion is not applicable for a subgradient algorithm[5], we are interested in the asymptotical convergence. Similar to [7], we define $\overline{\mu}^\alpha(T) := \frac{1}{T}\sum_{t=1}^T \mu^\alpha(t)$ and $\overline{\mu}^\beta(T) := \frac{1}{T}\sum_{t=1}^T \mu^\beta(t)$ as the average dual variables by time T, and let $\overline{\tilde{x}} := \frac{1}{T}\sum_{t=1}^T \tilde{x}(t)$ be the average primal variable by time T. The following theorems guarantee the statistical convergence of the subgradient method. The proofs are similar to [7], and are omitted due to the limit of space.

[5]This is because the dual cost usually will not monotonically approach the optimal value, but wander around it under the subgradient algorithm.

Theorem 4. Let $\mu^{\alpha *}$ and $\mu^{\beta *}$ be the optimal dual variables, then, for some $0 < B < \infty$, the following inequality holds

$$\limsup_{T\to\infty} D(\overline{\mu}^{\alpha}, \overline{\mu}^{\beta}) - D(\mu^{\alpha *}, \mu^{\beta *}) \leq hB. \tag{13}$$

Theorem 5. Let $\tilde{x}^*$ be the optimal rate of $\widetilde{\mathbf{P}}$, then, for some $0 < B < \infty$, the following inequality holds

$$\liminf_{T\to\infty} \widetilde{\mathbf{P}}(\overline{\tilde{x}}) \geq \widetilde{\mathbf{P}}(\tilde{x}^*) - hB. \tag{14}$$

The above theorems imply that the time-average primal and dual variables obtained by the subgradient algorithm can be made arbitrarily close to the optimal values if we choose the stepsize h sufficiently small.

7. DISTRIBUTED IMPLEMENTATION

In this section, we describe how the subgradient algorithm can be implemented in a real network in a distributed and scalable way. In our design, A source (aggregation) node needs to communicate only with its parent and children nodes. In detail, each node collects the transformed rate $\tilde{x}$ from its children, and updates the prices (μ^{α} and μ^{β}) based on Eqn. (11) and Eqn. (12). Then, it broadcasts updated prices to its children. Upon receiving the price information from its parent, each node calculates its transformed rate based on Eqn. (9) or Eqn. (10). Then, it forwards its updated rate to its parent. Moreover, the nodes solve the scheduling problem through the distributed algorithm as we discussed previously in Section 6.4, and decide who will have a chance to transmit in the next slot. Before convergence, each node transmits at a rate $\hat{x} = \min(x, r)$. At each subslot, it must conform to the data availability constraint.

In our algorithm, in each iteration, an independent set is picked as the solution of the scheduling problem. From a long-term perspective, the algorithm jumps among the extreme points (i.e., r^i) of the capacity region (recall that each extreme point corresponds to an independent set.), and never touches the inner area. As aforementioned, the optimal solution (i.e., r^*) is the convex combination of $|\mathcal{L}|$ extreme points (i.e., $r^* = \sum_{i=1}^{|\mathcal{L}|} \alpha_i r^i$), located on a facet of the capacity region's boundary. In reality, each α_i is actually the percentage of iterations that the algorithm picks the ith independent set, after the system converges.

8. DISCUSSIONS

8.1 Validity of Data Availability Constraint

As mentioned in Section 4.3, the data availability constraint is not taken into account when we solve the optimization problem $\mathbf{P}$. However, this will not cause any problem as long as the rate of each flow converges to a feasible point. In an aggregation node, it maintains a queue for each of its children, and one more queue for the available data. Suppose the packets in each queue are sorted by their timestamps, as shown in Fig. 1 and Fig. 2. Thus, the height of each queue is determined by the timestamp (i.e., T_t) of the packet on the top of this queue. If the aggregation node behaves strictly following the data availability constraint, the queue of the available data should have the same height as the shortest child queue. Clearly, after the optimal solution which satisfies both the network capacity constraint and data aggregation constraint is attained, the height of each child queue as well as the queue storing the available data will not grow infinitely.

Furthermore, our solution is suboptimal, and thus does not utilize the network resource to the extreme. Therefore, there is no doubt that the proposed scheme in this paper will not overflow any node in the aggregation tree.

8.2 Periodic Data Collection

Some sensing tasks require periodic data collection, namely, the intervals between the timestamps of consecutive packets are fixed. In this case, if we further assume synchronized data collection, i.e., all the sources start their collection at the same time, we can achieve the largest time-overlap of the packets, and thus maximize the rate of each source node. However, in practice, the time-offsets of the packets from different nodes may be time-varying, due to the dynamic join (leave) of sensor nodes, and the oscillation of the rates caused by the variation of the environment as well as the underlying MAC layer scheduling. In this scenario, the proposed algorithm can be considered as a good approximation, since the probabilistic rate model can capture the long-term expectation of time-offsets. For example, in the scenario shown in Fig. 1, node A and B both collect data in a periodic pattern, and their rates are $\frac{1}{2}$ and $\frac{1}{4}$. There are two possibilities for the time-offset of the packets from A and B, as shown in Fig. 1(b) and Fig. 1(c). The rate of aggregated packets (i.e., node C's sending rate) in these two cases are $\frac{1}{2}$ and $\frac{3}{4}$, respectively. If either case has the same chance to happen, the expected rate of the aggregation flow is $\frac{1}{2} \times (\frac{1}{2} + \frac{3}{4}) = \frac{5}{8}$. This exactly equals the lower bound of node C's rate derived by the data aggregation constraint $(1 - (1 - \frac{1}{2})(1 - \frac{1}{4}) = \frac{5}{8})$.

Even if the time-offsets can be controlled, however, in this scenario it is extremely difficult to mathematically model the relationship between a parent flow and its children flows in a convex function. Suppose the data collection are all synchronized, what we can do is to provide some tricks which can improve the objective value after the algorithm converges. In detail, we check the source flows sharing the same parent. If their periods are co-prime to each other, there is nothing can be improved since the data aggregation constraint precisely models the aggregation of the source flows with coprime periods. If the periods of some flows share a greatest common divisor α (let F^{α} be the set of them), we fix their rates as constants in $\mathbf{P}$, and use a virtual flow f_{α} to replace them in the data aggregation constraint. f_{α} is resulted from aggregating the flows in F^{α} when they are synchronized, and its rate is the constant $x_{f_\alpha} = \frac{1}{\alpha}(1 - \prod_{f\in F^{\alpha}}(1 - \alpha x_f))$. Subsequently, we restart the optimization of $\mathbf{P}$. Since x_{f_α} is lower than the rate derived by the data aggregation constraint, some network resources are saved, and thus the rates of other flows can be improved. As an example, suppose the rates of the flows from node A and B shown in Fig. 1 are $\frac{1}{4}$ and $\frac{1}{6}$, we have $x_{f_\alpha} = \frac{1}{3}$ according to above formula. Obviously, x_{f_α} is lower than the rate obtained based on Eqn (5), which is $\frac{3}{8}$. Thus, some bandwidth can be saved from the flow originated at node C and allocated to its neighboring flows. After reoptimization, the rates of these neighboring flows will be improved.

8.3 Lossy Link

Due to the unliable nature of wireless communication, packets may be lost during transmission. In our scheme, lost packets do not matter at all, since from the perspective of the receiver, lost packets look like "nonexistent packets", namely, the source nodes never collect data at those subslots. Furthermore, if the average reception probability of each link can be measured, the formulation of the problem can be easily redefined so as to take it into account. Retransmissions are not needed in our solution.

8.4 Energy Constraint

Energy scarcity is a major challenge for the design of sensor networks. Our approach can also be adapted to address this problem. The solution is to add an energy constraint to the problem formulation. As a result, the energy budget of each node on the aggregation

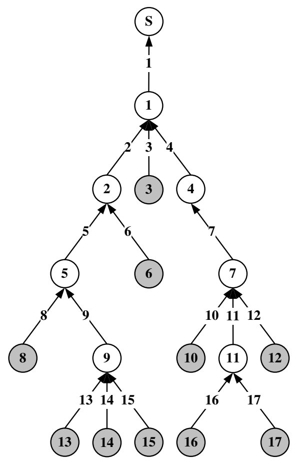
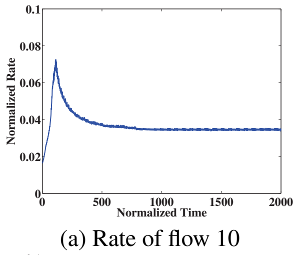
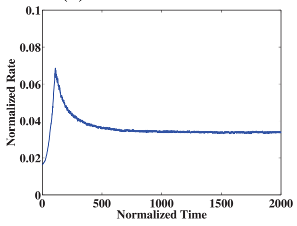


Figure 4: An aggregation tree



(a) Rate of flow 10



(b) Rate of flow 14

Figure 5: Rates of source flows

tree will be considered when the algorithm allocates the resource of the network.

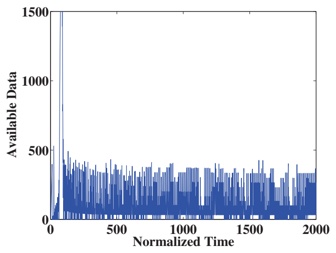
8.5 Time Synchronization

The problem of synchronization has been considered and addressed by the prior work on data aggregation in sensor networks [1], and we just borrow the existing solutions.

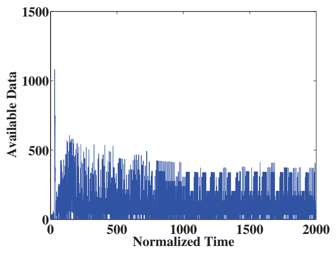
9. PERFORMANCE EVALUATION

In this section, we provide simulation results to complement the analysis in the previous sections. We consider a randomly generated aggregation tree shown in Fig. 4. On this tree, 10 source nodes (shaded nodes) collect and forward data to the sink S, through 7 aggregation nodes. In addition, the number on each node (edge) works as the index of this node (flow). First, we assume that all the links have a normalized capacity of 0.5, and all the source nodes use the same utility function $U(x) = \ln(x)$. Then, we apply our joint rate control and scheduling algorithm with a fixed stepsize $h = 1$ on this aggregation tree, and observe its performance.

Figure 5 shows the evolution of the rates of the source flow 10 and 14. The other source flows have similar behavior and thus we omit their results. As one can see, they converge quickly to a neighborhood of the optimal values and oscillate around the optimal values. This oscillating behavior mathematically results from the nondifferentiability of the dual function and physically can be interpreted as due to the scheduling process.



(a) Available data of node 5



(b) Available data of node 7

Figure 6: Available data stored in aggregation nodes

Figure 6 describes the amount of available data stored in node 5 and 7. The other nodes have similar behavior and thus we omit their results. In this test, we assume each time slot (length of step) contains 100 subslots. As can be seen, although the two curves both fluctuate with the time going, they are bounded reasonably. The rise of fluctuation can be ascribed to the underlying scheduling, which prevents an aggregation node from receiving and transmiting packets at the same time.

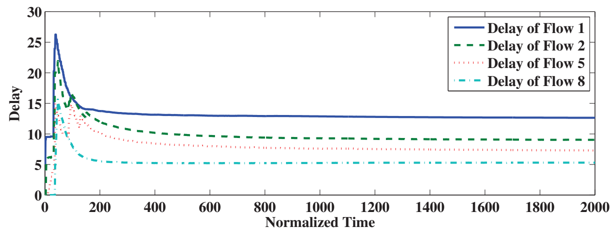
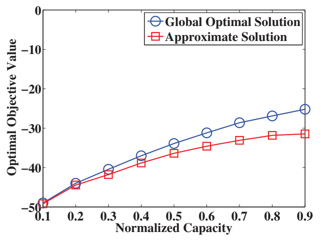
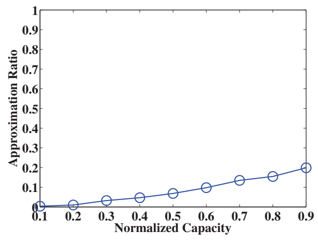


Figure 7: Delays of the packets delivered by flows

Figure 7 demonstrates the average delays of the packets delivered by four flows. Here the concept of delay is defined as the period from the moment when a packet is generated by a source to the time when this packet is delivered by a flow. For example, suppose node 8 generates a packet at time 1. Based on the delay values shown in Fig. 7, this packet will arrive at node 5 at time 6, since the delay of flow 8 is roughly 5. Similarly, it will arrive at node 2 at time 8 since the delay of flow 5 is 7, and node 1 at time 10 since the delay of flow 2 is 9. Finally it will reach the sink at time 14 since the delay of flow 1 is 13. Here we should note that this packet may be aggregated during this process. As can be seen in the figure, the average delay of each flow converges to a stable point soon after the algorithm is started.



(a) Optimal Value



(b) Approximation Ratio

Figure 8: Optimal Solution V.S. Approximate Solution

Finally, Fig. 8 discloses the difference between the optimal objective value and the approximate objective value. In this test, we tune the capacity of each link, and observe its impact on the objective value. Intuitively, as illustrated in Fig. 8(a), both the optimal objective value and the approximate objective value increase with the capacities growing. At first, when the capacity of each link is as low as 0.1, the difference between the two values is negligible, since the approximation ratio defined as $\frac{|\text{Optimal}-\text{Approximate}|}{|\text{Approximate}|}$ is less than 1%, as shown in Fig. 8(b). As the capacities grow up, the difference as well as the approximation ratio increase accordingly. They remain in a low level (less than 10%) until the capacity goes beyond 0.5. Finally, the approximation ratio reaches around 20% when the capacity is increased to 0.9.

To find a reasonable explanation for this point, let us observe the function we use in all the variable substitutions, which is $f(x) = -\ln(1-x)$. The curvature, i.e., the second order derivative of this function is monotonously increasing with x. Thus, when the original variable x is small, the value of the new variable $y = f(x)$ is close to x. For this reason, when we decrease the capacity of each link, the boundary of the transformed region Π' on which the approximate solution (i.e., $\tilde{r}_0$ shown in Fig. 3(c)) is located will become closer to the boundary of the approximate region $\widetilde{\Pi}$ on which the optimal solution (i.e., $\tilde{r}^*$ shown in Fig. 3(c)) is located. Consequently, $\tilde{r}^* - \tilde{r}_0$ drops accordingly, resulting in a smaller difference between the optimal value and the approximate value, by Theorem 3. Finally, it should be noted that the optimal value shown in this test is actually the optimal value of the approximate problem (i.e., $\widetilde{\mathbf{P}}^*$), which has been proved to be the upper bound of the real optimal value $\mathbf{P}^*$. This implies that in practice, our approximate solution should be even closer to the real optimal solution than we observe in this experiment.

10. CONCLUSIONS

In this paper, we identify the unique challenges of rate allocation in the context of data aggregation in wireless sensor networks, and formulate this problem as a network utility maximization problem. After transforming this problem into a convex approximate problem, we decompose it based on the duality theory, and propose a distributed algorithm to solve the decoupled problems. Theoretical analysis and simulation results demonstrate the near-optimal performance of our scheme.

11. ACKNOWLEDGMENTS

We are grateful to Professor P.R.Kumar for his kind help on the formulation and solution of the problem. Thanks are also due to Prof.Angelia Nedich, Prof.Tarek Abdelzaher, I-Hong Hou and Yuhan Zhang for providing helpful comments and feedback.

12. REFERENCES

[1] S. M. Michael, M. J. Franklin, J. Hellerstein, and W. Hong, “Tag: a tiny aggregation service for ad-hoc sensor networks,” in *OSDI*, 2002.

[2] F. Kelly, “Charging and rate control for elastic traffic,” *European Transactions on Telecommunications*, vol. 8, pp. 33–37, 1997.

[3] F. Kelly, A. Maulloo, and D. Tan, “Rate control in communication networks: shadow prices, proportional fairness and stability,” in *Journal of the Operational Research Society*, vol. 49, 1998.

[4] S. H. Low and D. E. Lapsley, “Optimization flow control-i: basic algorithm and convergence,” *IEEE/ACM Trans. Netw.*, vol. 7, no. 6, pp. 861–874, 1999.

[5] K. Kar, S. Sarkar, and L. Tassiulas, “Optimization based rate control for multirate multicast sessions,” in *INFOCOM*, 2001.

[6] Y. Cui, Y. Xue, and K. Nahrstedt, “Optimal resource allocation in overlay multicast,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 17, no. 8, pp. 808–823, Aug. 2006.

[7] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle, “Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks,” in *INFOCOM*, 2006.

[8] X. Lin and N. B. Shroff, “Joint rate control and scheduling in multihop wireless networks,” in *CDC*, 2004.

[9] X. Lin, N. Shroff, and R. Srikant, “A tutorial on cross-layer optimization in wireless networks,” *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 8, pp. 1452–1463, Aug. 2006.

[10] A. Eryilmaz and R. Srikant, “Joint congestion control, routing, and mac for stability and fairness in wireless networks,” *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 8, pp. 1514–1524, 2006.

[11] P.-J. Wan, K. Alzoubi, and O. Frieder, “Distributed construction of connected dominating set in wireless ad hoc networks,” in *INFOCOM*, 2002.

[12] 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 *INFOCOM*, 2008.

[13] B. Yu, J. Li, and Y. Li, “Distributed data aggregation scheduling in wireless sensor networks,” in *INFOCOM*, 2009.

[14] P.-J. Wan, S. C.-H. Huang, L. Wang, Z. Wan, and X. Jia, “Minimum-latency aggregation scheduling in multihop wireless networks,” in *MobiHoc*, 2009.

[15] Y. T. Hou, Y. Shi, and H. D. Sherali, “Rate allocation and network lifetime problems for wireless sensor networks,” *IEEE/ACM Trans. Netw.*, vol. 16, no. 2, pp. 321–334, 2008.

[16] Y. Xue, Y. Cui, and K. Nahrstedt, “Maximizing lifetime for data aggregation in wireless sensor networks,” *MONET*, vol. 10, no. 6, pp. 853–864, 2005.

[17] B. Przydatek, D. X. Song, and A. Perrig, “Sia: secure information aggregation in sensor networks,” in *SenSys*, 2003.

[18] Y. Yang, X. Wang, S. Zhu, and G. Cao, “Sdap: a secure hop-by-hop data aggregation protocol for sensor networks,” in *MobiHoc*, 2006.

[19] A. Giridhar and P. R. Kumar, “Toward a theory of in-network computation in wireless sensor networks,” *IEEE Communications Magazine*, vol. 44, pp. 98–107, 2006.

[20] D. Marco, E. Duarte-Melo, M. Liu, and D. Neuhoff, “On the many-to-one transport capacity of a dense wireless sensor network and the compressibility of its data,” in *IPSN*, 2003.

[21] S. Chen and Z. Zhang, “Localized algorithm for aggregate fairness in wireless sensor networks,” in *MOBICOM*, 2006.

[22] C. T. Ee and R. Bajcsy, “Congestion control and fairness for many-to-one routing in sensor networks,” in *SenSys*, 2004.

[23] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, “Impact of interference on multi-hop wireless network performance,” in *MOBICOM*, 2003.

[24] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[25] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1995.

[26] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, 1998.

[27] J.-H. Hoepman, “Simple distributed weighted matchings,” *CoRR*, 2004.

[28] L. Su, C. Liu, H. Song, and G. Cao, “Routing in intermittently connected sensor networks,” in *ICNP*, 2008.

[29] L. Su, B. Ding, Y. Yang, T. F. Abdelzaher, G. Cao, and J. C. Hou, “ocast: Optimal multicast routing protocol for wireless sensor networks,” in *ICNP*, 2009.

[30] N. Z. Shor, K. C. Kiwiel, and A. Ruszcayn̂ski, *Minimization methods for non-differentiable functions*. New York, NY, USA: Springer-Verlag New York, Inc., 1985.