Event-triggered distributed optimization in sensor networks

Pu Wan and Michael D. Lemmon [*]

ABSTRACT

Many problems in sensor networks can be formulated as optimization problems. When distributed optimization algorithms are used, sensor nodes communicate with each other in order to collaboratively solve the overall network optimization problem. Existing distributed optimization algorithms typically rely on choosing a step size to ensure the convergence of the algorithm. In this case, the communication between sensor nodes occurs each time the computations are carried out. This is highly undesirable since the choice of step size is usually small, which means the number of messages exchanged between nodes will be large. Since in sensor networks, the energy required for communication can be significantly greater than the energy required to perform computation, it would be beneficial if we can somehow separate communication and computation. This paper presents such an distributed algorithm called the event-triggered algorithm. Under event triggering, each agent broadcasts to its neighbors when a local "error" signal exceeds a state dependent threshold. We give a general class of optimization problems in sensor networks where the event-triggered algorithm can be used. In particular, this paper uses the data gathering problem as an example. We propose an event-triggered distributed algorithm for the data gathering problem and prove its convergence. In the simulation, we assume tree communication structure, and all leaf nodes and intermediate nodes send data to the root node (sink node). Simulation results show that the proposed algorithm reduces the number of message exchanges by two orders of magnitude compared to commonly used dual decomposition algorithms. It also enjoys better scalability with respect to the depth of the tree and the maximum branch number of the tree.

[*]Both authors are with the department of Electrical Engineering, Univ. of Notre Dame, Notre Dame, IN 46556; e-mail: pwan,lemmon@nd.edu. The authors gratefully acknowledge the partial financial support of the National Science Foundation (NSF-ECS04-00479 and NSF-CNS-07-20457).

1. INTRODUCTION

Wireless sensor networks consist of microprocessor controlled sensors that communicate with each other over multi-hop communication networks. Many problems in wireless sensor networks, including estimation [1] [2] [3] [4], source localization [1], data gathering [5] [6], maximum lifetime routing [7], control [8], resource allocation [9] [10], and congestion control [11][12], can be formulated as optimization problems. When distributed optimization algorithms are used to solve the above problems, sensor nodes communicate with each other in order to collaboratively solve the overall network optimization problem.

Existing distributed optimization algorithms, like dual decomposition [13], typically rely on choosing a step size to ensure the convergence of the algorithm. In this case, the communication between sensor nodes occurs each time the computations are carried out at the sensor nodes. This is highly undesirable since the choice of step size is usually small, which means the number of messages exchanged between sensor nodes will be large. Since in sensor networks, the energy required for communication can be significantly greater than the energy required to perform computation [14]. As a result, it would be beneficial if we can somehow separate communication and computation in distributed algorithms. This should reduce the message passing complexity of distributed optimization algorithms such as dual decomposition significantly.

This paper presents one way of reducing the message passing complexity by using a new class of distributed algorithms called the event-triggered algorithm. Under event triggering, each agent broadcasts to its neighbors when a local "error" signal exceeds a state dependent threshold. We first give a general framework of optimization problems in sensor networks where the event-triggered algorithm can be used. Three specific applications in sensor networks, data gathering, estimation/localization, and maximum lifetime routing are then shown to fall into this general framework. The rest of the paper uses the data gathering problem as an illustrating example to show the idea and effectiveness of the event-triggered algorithm. We propose an event-triggered distributed algorithm for the data gathering problem and prove its convergence. In the simulation, we assume tree communication structure, and all leaf nodes and intermediate nodes send data to the root node (sink node). Simulation results show that the proposed algorithm reduces the number of message exchanges by two orders of magnitude compared to commonly used dual decomposition algorithms. It also enjoys better scalability with respect to the depth of the tree and the maximum branch number of the tree.

The rest of the paper is organized as follows. Section 2 gives the general framework of optimization problems and three specific applications in sensor networks. Section 3 reviews the commonly used dual decomposition algorithm. The event-triggered optimization algorithm is based on an augmented Lagrangian methods, which is described in section 4. Section 5 presents our event-triggered distributed algorithm based on the augmented Lagrangian methods, and proves its convergence. Simulation results are shown in section 6, and section 7 concludes the paper.

2. PROBLEM FORMULATION AND APPLICATIONS IN SENSOR NETWORKS

Many problems in wireless sensor networks can be formulated as an optimization problem of the following general form:

$$\begin{array}{rl} \text{minimize:} & f(x) = \sum_{i\in\mathcal{V}} f_i(x_i, y_i) \\ \text{w.r.t:} & x \\ \text{subject to:} & \mathcal{A}x \le g \end{array} \qquad (1)$$

Here $\mathcal{V} = \{1, \cdots, N\}$ is the set of sensor nodes, x_i is some variable/state local to node i, and y_i is some data/measurement local to node i. $x = [x_1, \cdots, x_N]^T$. $f_i(x_i, y_i)$ is a local cost function associated with node i, which depends on both x_i and y_i. $\mathcal{A}$ is some sparse matrix characterizing the local interconnection structure of the sensor network, and $\mathcal{A}x \le g$ represents some local coupling constraints or balance the sensor network must respect. The objective of the problem is to find such local states of the nodes such that the total cost of the network is minimized, subject to local linear constraints.

We will next present three classes of problems in sensor networks that fit into this framework, the data gathering problem, the estimation/localization problem, and the maximum lifetime routing problem. We will use the data gathering problem later to show the idea of event-triggered optimization.

2.1 Data gathering

The data gathering problem we consider here is from [5] [6]. Consider a network of wireless sensor nodes that are distributed over a region. Each sensor node has limited battery and can sense certain data in the region that needs to be communicated to a sink node (base station). We assume that each node also has the capability to relay packets. Chen et al [5] [6] associated a utility function to each sensor node, which represents the amount of useful information in the data. Instead of maximizing the total amount of raw data collected at the sink node, we are interested in maximizing the total amount of useful information communicated to the sink node subject to node capacity and energy constraints.

The data gathering problem above can be formulated as a network utility maximization (NUM) problem [15]. Here we assume that the sensor network is continuously gathering data, and the energy required to do local computation is negligible compared to the energy required for communication [14]. Each node generates a flow with a specified data rate that goes to the sink. Each flow may traverse several hops (which together constitute a route) before reaching the sink node. Here each sensor node both generates traffic and relays traffic for other nodes, which means it works as both a user and a link. The set of nodes that relays traffic for node $i \in \mathcal{V}$ is denoted as $\mathcal{L}_i$ and the set of nodes that use node $i \in \mathcal{V}$ as a relay node is denoted as $\mathcal{S}_i$. It is shown in [6] that the data gathering problem can be formulated as

$$\begin{array}{rl} \text{maximize:} & U(x) = \sum_{i\in\mathcal{V}} U_i(x_i) \\ \text{w.r.t:} & x \\ \text{subject to:} & Ax \le \overline{c}, \quad x \ge 0 \\ & (e_{tx} + e_{rx})Ax - e_{rx}x \le b(t) \end{array} \qquad (2)$$

Here $x = [x_1, ..., x_N]^T$ and $x_i \in \mathbb{R}$ is node i's data rate. $\overline{c} \in \mathbb{R}^M$ is a vector of node capacities (the maximum data rate each node can transmit, in bps). $A \in \mathbb{R}^{N\times N}$ is the routing matrix defining the relaying relationship between sensor nodes. The ji'th component, A_{ji}, is 1 if node j relays traffic for node i and is zero otherwise. The jth row of Ax represents the total data rates node j needs to transmit, which cannot exceed its capacity $\overline{c}_j$. In the second inequality constraint, e_{tx} and e_{rx} represent the energy consumed in transmitting and receiving one unit of data, respectively. $b_i(t)$ is the intended energy consumption rate for node i at time t, which is defined as

$$b_i(t) = E_i(t)/(T_l - t) \tag{3}$$

Here $E_i(t)$ is the remaining energy of node i at time t, and T_l is the pre-specified desired system lifetime of the sensor network. $b(t) = [b_1(t), \cdots, b_N(t)]^T$. So the second inequality constraint in problem 2 basically says that each node needs to control the data rates so that its energy consumption rate is below the rate that sustains the specified system lifetime. The cost function U is the sum of the node *utility* functions $U_i(x_i)$. In the Internet context, these utility functions represent the reward (i.e. quality-of-service) [15][13] user i gets by transmitting at rate x_i. Here in the data gathering problem, it represents the amount of useful information in node i's data flow. Problem 2 tries to maximize the total amount of useful information gathered at the sink node subject to capacity and energy constraints. Notice that $b(t)$ is a function of time, which makes problem 2 time-varying in nature.

The capacity and energy constraints in problem 2 can be combined to one single linear inequality constraint of the following form

$$\begin{bmatrix} A \\ A - \frac{e_{rx}}{e_{tx}+e_{rx}} I \end{bmatrix} x \leq \begin{bmatrix} \overline{c} \\ \frac{b(t)}{e_{tx}+e_{rx}} \end{bmatrix} \tag{4}$$

This transforms problem 2 into the general form of problem 1. We can thus develop an event-triggered algorithm for problem 2.

We use the first order radio model in [14] to model the energy consumption in our data gathering problem. In this case

$$\begin{aligned} e_{tx} &= E_{elec} + \epsilon_{amp} \times d^2 & (5)\\ e_{rx} &= E_{elec} & (6) \end{aligned}$$

where d is the transmission distance between two nodes. The tranmitter/receiver electronics $E_{elec} = 50nJ/bit$, and transmitter amplifier $\epsilon_{amp} = 100pJ/bit/m^2$. If we consider a transmission distance of $d = 200m$, then $e_{tx} = 81e_{rx} \gg e_{rx}$. This means in this case we can ignore the energy consumption associated with receiving packets. The sensor networks that have such transmission distances are present in [16] [17]. *We should emphasize that, the above assumption is not needed for developing the event-triggered algorithm for problem 2. However, it simplifies the notations and descriptions of the algorithm in section 5 significantly.*

When $e_{tx} \gg e_{rx}$, the constraint in equation 4 is the same as

$$Ax \leq c(t) = \min\left\{\overline{c}, \frac{b(t)}{e_{tx}+e_{rx}}\right\} \tag{7}$$

The data gathering problem 2 then reduces to

$$\begin{array}{rl} \text{maximize:} & U(x) = \sum_{i\in\mathcal{V}} U_i(x_i) \\ \text{w.r.t:} & x \\ \text{subject to:} & Ax \leq c(t) = \min\left\{\overline{c}, \frac{b(t)}{e_{tx}+e_{rx}}\right\}, \quad x \geq 0 \end{array} \tag{8}$$

Problem 8 is a standard Network Utility Maximization (NUM) problem [15][13] if $c(t)$ is constant for all t. NUM problems are usually solved using dual decomposition algorithm [13]. We will review the dual decomposition algorithm in section 3, and present our event-triggered distributed algorithm for problem 8 in section 5.

2.2 Estimation/Localization

Many estimation and localization problems in sensor networks fit into the general framework of problem 1. Take the estimation problem for example, the objective is to estimate some parameter θ over a sensor field. y_i is node i's local measurement of the parameter. The estimation problem can usually be cast as an optimization problem in the following form:

$$\begin{array}{rl} \text{minimize:} & \sum_{i\in\mathcal{V}} f_i(\theta, y_i) \\ \text{w.r.t:} & \theta \end{array} \tag{9}$$

For the basic least square estimation problem, we have $f_i(\theta, y_i) = \frac{1}{2}(\theta - y_i)^2$. In the robust estimation problem [1], the Huber loss function in equation 10 is used.

$$f_i(\theta, y_i) = \begin{cases} \frac{1}{2}(\theta - y_i)^2, & \text{if} \quad |\theta - y_i| \leq \beta \\ \beta(\theta - y_i) - \beta^2/2, & \text{if} \quad |\theta - y_i| > \beta \end{cases} \tag{10}$$

Here β is some preset constant.

The source localization problem in [1] can also be represented by problem 9. In that particular problem, an acoustic source is placed at an unknown location, θ, in the sensor field. The source emits isotropically a signal, and we would like to estimate the location of the source using received signal energy measurements taken at each sensor. Assume each sensor node knows its own location, r_i, relative to a fixed reference point. Then from the energy propagation model, the received signal energy measurement at node i will be

$$y_i = \frac{\mathcal{H}}{|\theta - r_i|^\alpha} + \varepsilon_i \tag{11}$$

where $\mathcal{H}, \alpha$ are constants, and ε_i is some measurement noise. In this case the cost function in problem 9 will be

$$f_i(\theta, y_i) = \left(y_i - \frac{\mathcal{H}}{|\theta - r_i|^\alpha}\right)^2 \tag{12}$$

To see how problem 9 fits into the general framework of problem 1, we need to introduce a local copy of θ at each node. To be specific, we let x_i be node i's estimate of θ. In this way, problem 9 can be transformed to the following problem:

$$\begin{array}{ll} \text{minimize:} & \sum_{i\in\mathcal{V}} f_i(x_i, y_i) \\ \text{w.r.t:} & x \\ \text{subject to:} & \mathcal{I}x = 0 \end{array} \tag{13}$$

Here $\mathcal{I}$ is the incidence matrix of the communication graph associated with the sensor network. The equality constraint in problem 13 basically says $x_i = x_j$, $\forall i, j \in \mathcal{V}$. This means each node's local estimate of θ should asymptotically converge to the same value.

2.3 Maximum lifetime routing

Another example that can be fit into the general optimization framework is the maximum lifetime routing problem in sensor networks. [7] considered the problem of computing an optimal routing scheme that maximizes the time at which the first node in the sensor network drains out of energy. The problem was formulated as an convex optimization problem, and dual decomposition was then used to develop a distributed algorithm.

Let an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ denote the sensor network, where $\mathcal{V}$ is the set of nodes, and $\mathcal{E}$ is the set of links. $N = |\mathcal{V}|$, $M = |\mathcal{E}|$. Two nodes are connected by a link if they can transmit a packet to each other with a transmission power less than the maximum transmission power at each node. All links are assumed to be bi-directional. Let $\mathcal{N}_i$ denote the set of nodes connected to node i by a link. Each node i is assumed to have an initial battery energy E_i, and R_i is the rate at which information is generated at node i. All this information needs to be communicated to a sink node. Let r_{ij} denote the flow rate from node i to node j, and assume that the energy spent by node i to transmit a unit of information to node j is e_{ij}. It is shown in [7] that the maximum lifetime routing problem can be formulated as

$$\begin{array}{ll} \text{minimize:} & \sum_{i\in\mathcal{V}} q_i^2 \\ \text{w.r.t:} & q, r \\ \text{subject to:} & \sum_{j\in\mathcal{N}_i} (r_{ij} - r_{ji}) = R_i, \quad \forall i \in \mathcal{V} \\ & r_{ij} \geq 0, \quad \forall i \in \mathcal{V}, \forall j \in \mathcal{N}_i \\ & \sum_{j\in\mathcal{N}_i} e_{ij} r_{ij} \leq q_i E_i, \quad \forall i \in \mathcal{V} \\ & q_i = q_j, \quad \forall i \in \mathcal{V}, \forall j \in \mathcal{N}_i \end{array} \tag{14}$$

Here q_i is a local variable at node i, which represents node i's estimate of the reciprocal of the maximum lifetime. The first inequality in problem 14 is the flow balance equation at node i, and the second inequality is simply a non-negativity constraint on the flow rate. The third constraint is an energy constraint at node i, and the fourth constraint simply requires that all local estimate of the maximum lifetime be equal.

It is easy to see that problem 14 can be reformulated as:

$$\begin{array}{ll} \text{minimize:} & \sum_{i\in\mathcal{V}} q_i^2 \\ \text{w.r.t:} & q, r \\ \text{subject to:} & A_1\mathbf{r} = \mathbf{R} \\ & \mathbf{r} \geq 0 \\ & \mathbf{er} - \mathbf{Eq} \leq 0 \\ & A_2\mathbf{q} = 0 \end{array} \tag{15}$$

where $\mathbf{r} = [r_1, \cdots, r_M]^T$, $\mathbf{R} = [R_1, \cdots, R_N]^T$, $\mathbf{q} = [q_1, \cdots, q_N]^T$. A_1, e, E, A_2 are appropriately defined constant sparse matrices depending on the topology of the graph. In problem 15, if we view $\mathbf{x} = \begin{bmatrix} \mathbf{r} \\ \mathbf{q} \end{bmatrix}$ as the state, then the problem is a convex optimization problem with linear equality and inequality constraints. It falls into the general framework of problem 1 as well.

So far we have presented a general framework of optimization problems, and three specific applications in sensor networks. In the rest of the paper, we will only focus on the data gathering problem 8, and use it as an illustrating example of event-triggered optimization.

3. DUAL DECOMPOSITION ALGORITHM

In this section we will review the dual decomposition algorithm in [13], which is the most used algorithm for solving the NUM problem. Later in section 6 we will compare the performance of our event-triggered algorithm against the dual decomposition algorithm.

A variety of distributed algorithms have been proposed to solve the NUM problem [15] [13] [18] [19]. Kelly [15] first proposed two classes of algorithms by decomposing the NUM problem into a user problem and a network problem. Among all existing algorithms, the dual decomposition approach proposed by Low et al. [13] is the most widely used algorithm for the NUM problem.

Assume $c(t) = c$ for all t. The algorithm solves the NUM problem in the form of problem 8 by examining the dual of the problem, which is

$$\begin{array}{ll} \text{minimize:} & \max_{x\geq 0} \left\{ \sum_{i\in\mathcal{V}} U_i(x_i) - p^T(Ax - c) \right\} \\ \text{subject to:} & p \geq 0 \end{array} \tag{16}$$

where $p = \begin{bmatrix} p_1 & \cdots & p_N \end{bmatrix}^T$ is the Lagrange multiplier vector (which can be viewed as the price for using each node for relaying traffic) associated with the inequality constraint $Ax \leq c$. If x^* and p^* are vectors solving the dual problem, then it can be shown that x^* also solves the original NUM problem.

Low et al. [13] established conditions under which a pair of recursions would generate a sequence of data rates, $\{x[k]\}_{k=0}^{\infty}$, and node prices, $\{p[k]\}_{k=0}^{\infty}$, that asymptotically converge to a solution of the dual problem. Given the initial data rates $x[0]$ and node prices $p[0]$, then for all $i \in \mathcal{V}$, we let

$$x_i[k+1] = \arg\max_{x_i \geq 0} \left\{ U_i(x_i[k]) - x_i[k] \sum_{j \in \mathcal{L}_i} p_j[k] \right\} \quad (17)$$

$$p_i[k+1] = \max \left\{ 0, p_i[k] + \gamma \left\{ \sum_{j \in \mathcal{S}_i} x_j[k] - c_i \right\} \right\} \quad (18)$$

for $k = 0, \cdots, \infty$.

The step size γ in equation 18 must be chosen to ensure that the sequences $\{x[k]\}_{k=0}^{\infty}$ and $\{p[k]\}_{k=0}^{\infty}$ asymptotically converge to the optimal solution. Low et al. [13] showed that a suitable step size is

$$0 < \gamma < \gamma^* = \frac{-2 \max_{(i,x_i)} \nabla^2 U_i(x_i)}{\overline{L}\,\overline{S}} \quad (19)$$

where $\overline{L}$ is the maximum number of relay nodes any node uses and $\overline{S}$ is the maximum number of nodes any node relays traffic for. Equation 19 requires that the step size be inversely proportional to both $\overline{L}$ and $\overline{S}$. We can conclude that the computational complexity of dual decomposition (as measured by the number of algorithm updates) scales superlinearly with $\overline{L}$ and $\overline{S}$.

Under dual decomposition, system agents exchange information at each iteration, so that step size also determines the message passing complexity of the algorithm. Therefore if we use the "stabilizing" step size, dual decomposition will have a message complexity that scales in a super-linear manner with $\overline{L}$ and $\overline{S}$.

4. AUGMENTED LAGRANGIAN METHOD NUM ALGORITHM

The event-triggered algorithm presented in this paper is based on the augmented Lagrangian method for the NUM problem. In the augmented Lagrangian method, a constrained problem is converted into a sequence of unconstrained problems by adding to the cost function a penalty term that prescribes a high cost to infeasible points.

The discussion in this section and next section also assumes $c(t) = c$ for all t. This assumption ensures that problem 8 has a well-defined unique solution. To apply the augmented Lagrangian method on our NUM problem 8, we need to introduce the slack variable $s \in \mathbb{R}^N$ and replace the inequalities $c_j - a_j^T x \geq 0$, $\forall j \in \mathcal{V}$ by

$$a_j^T x - c_j + s_j = 0, \quad s_j \geq 0, \quad \forall j \in \mathcal{V} \quad (20)$$

The augmented cost is then

$$\overline{L}(x, s; \lambda, w) = -\sum_{i \in \mathcal{V}} U_i(x_i) + \sum_{j \in \mathcal{V}} \lambda_j (a_j^T x - c_j + s_j) + \frac{1}{2} \sum_{j \in \mathcal{V}} \frac{1}{w_j} (a_j^T x - c_j + s_j)^2 \quad (21)$$

Here a penalty parameter w_j is associated with each constraint, and $w = [w_1, \cdots, w_N]$ is the vector of penalty parameters. Suppose λ_j^* is the Lagrange multiplier associated with node j's constraint $c_j - a_j^T x \geq 0$ in the Karush-Kuhn-Tucker conditions of the NUM problem. λ_j is an estimate of λ_j^* and $\lambda = [\lambda_1, \cdots, \lambda_N]$. The vector $a_j^T = [A_{j1}, \cdots, A_{jN}]$ is the jth row of the routing matrix A.

$\overline{L}(x, s; \lambda, w)$ is a continuous function of x and s for fixed λ and w. It is shown [20] that

$$\min_{x \geq 0, s \geq 0} \overline{L}(x, s; \lambda, w) = \min_{x \geq 0} \min_{s \geq 0} \overline{L}(x, s; \lambda, w) = \min_{x \geq 0} L(x; \lambda, w)$$

where *the augmented Lagrangian function* associated with the NUM problem is given as

$$L(x; \lambda, w) = -\sum_{i \in \mathcal{V}} U_i(x_i) + \sum_{j \in \mathcal{V}} \psi_j(x; \lambda, w) \quad (22)$$

where

$$\psi_j(x; \lambda, w) = \begin{cases} -\frac{1}{2} w_j \lambda_j^2, \text{if} \quad c_j - a_j^T x - w_j \lambda_j \geq 0 \\ \lambda_j (a_j^T x - c_j) + \frac{1}{2w_j} (a_j^T x - c_j)^2, \text{otherwise} \end{cases}$$

The augmented Lagrangian method solves the NUM problem by minimizing $L(x; \lambda[k], \overline{w}[k])$ for sequences of $\{\overline{w}[k]\}_{k=0}^{\infty}$ and $\{\lambda[k]\}_{k=0}^{\infty}$. Let $x^*[k]$ denote the approximate minimizer for $L(x; \lambda[k], \overline{w}[k])$. The method in [20, Chap 4.2] can be used to show that for appropriately chosen sequences $\{\overline{w}[k]\}_{k=0}^{\infty}$ and $\{\lambda[k]\}_{k=0}^{\infty}$, the sequence of approximate minimizers $\{x^*[k]\}_{k=0}^{\infty}$ converges to the optimal point of the NUM problem. The choices are as follows. $\{\overline{w}_j[k]\}_{k=0}^{\infty}$ are sequences of penalty parameters that are monotone decreasing to zero. $\{\lambda_j[k]\}_{k=0}^{\infty}$ are sequences of Lagrange multiplier estimates, where $\lambda_j[k+1] = \max\{0, \lambda_j[k] + \frac{1}{\overline{w}_j[k]}(a_j^T x^*[k] - c_j)\}$.

In our earlier work in [21], we gave an augmented Lagrangian method algorithm for the NUM problem that converges to the exact minimizer of the NUM problem. Since in sensor networks, sensor nodes have very limited battery life, and it usually suffices to obtain an approximate minimizer to the problem. So instead of minimizing $L(x; \lambda[k], \overline{w}[k])$ for sequences of $\{\overline{w}[k]\}_{k=0}^{\infty}$ and $\{\lambda[k]\}_{k=0}^{\infty}$, we are only considering the problem of minimizing $L(x; \lambda, w)$ for fixed λ and w in this paper. If $\lambda_j = 0$ and w_j is sufficiently small, the minimizer of $L(x; \lambda, w)$ will be a good approximation to the solution of the original NUM problem. The algorithm is given as follows:

1. **Initialization:** Select any initial data rate $x^0 > 0$, Set $\lambda_j = 0$ and sufficiently small $w_j > 0$, $j \in \mathcal{V}$.

2. **Recursive Loop: Minimize** $L(x; \lambda, w)$

$$x = \max\left\{0, x^0 - \gamma \nabla_x L(x^0; \lambda, w)\right\} \quad (23)$$
$$x^0 = x$$

The above algorithm converges to an approximate solution of the original NUM problem. The smaller w is, the more accurate the approximation is. The recursion shown in step 2 is minimizing $L(x; \lambda, w)$ using a simple gradient following method. γ is a sufficiently small step size.

The computations above can be easily distributed among the sensor nodes. We will see how they are distributed in our event-triggered distributed implementation of the algorithm in section 5.

In dual decomposition and the algorithm shown above, the exchange of information between the sensor nodes happens each time the gradient following update is applied. This means that the number of messages passed between the nodes is equal to the number of updates required for the algorithm's convergence. That number is determined by the step size. For both algorithms, these step sizes may be small, so that the number of messages passed will be large.

The message passing complexity can be greatly reduced by using event-triggered messages. The following section presents an event-triggered distributed implementation of the algorithm presented in this section.

5. EVENT-TRIGGERED DISTRIBUTED OPTIMIZATION IN SENSOR NETWORKS

Implementing the algorithm in section 4 in a distributed manner requires communication between the sensor nodes. An event-triggered implementation of the algorithm assumes that the transmission of messages is triggered by some local error signal crossing a state-dependent threshold. The main problem is to determine a threshold condition that results in message streams ensuring the asymptotic convergence of the algorithm to the NUM problem's approximate solution. This section determines such an event threshold condition and gives an distributed algorithm for solving problem 8.

For the data gathering problem we considered, each sensor node both generates traffic and relays traffic for other nodes, which means it works as both a user and a link. Recall that the set of nodes that relays traffic for node $i \in \mathcal{V}$ is denoted as $\mathcal{L}_i$ and the set of nodes that use node $i \in \mathcal{V}$ as a relay node is denoted as $\mathcal{S}_i$.

We can search for the minimizer of the Lagrangian $L(x; \lambda, w)$ using a gradient following algorithm

$$\begin{aligned} x_i(t) &= -\int_0^t \left(\nabla_{x_i} L(x(s); \lambda, w)\right)^+_{x_i(s)} ds \\ &= \int_0^t \left(\frac{\partial U_i(x_i(s))}{\partial x_i} - \sum_{j \in \mathcal{L}_i} \mu_j(s) \right)^+_{x_i(s)} ds \end{aligned} \quad (24)$$

for each node $i \in \mathcal{V}$ and where

$$\mu_j(t) = \max\left\{0, \frac{1}{w_j}(a_j^T x(t) - c_j)\right\} \quad (25)$$

Here given a function $f : \mathbb{R}_+ \to \mathbb{R}$, its *positive projection* is defined as

$$(f(x))^+_x = \begin{cases} 0, & \text{if} \quad x = 0 \quad \text{and} \quad f(x) < 0 \\ f(x), & \text{otherwise} \end{cases} \quad (26)$$

The positive projection used in equation 24 guarantees the data rate $x_i(t)$ is always nonnegative along the trajectory.

Equation 24 is the continuous-time version of the update in equation 23. Note that in equation 24, node i can compute its rate only based on the information from itself, and the information of μ_j from those nodes that relay traffic for node i. We can think of μ_j as the jth node's *link state*, which is local to node j. From equation 25, node j only needs to be able to measure the total flow that goes through itself. All of this information is locally available so the update of the data rate can be done in a distributed manner.

In the above equation, this link state information is available to other nodes in a continuous manner. We now consider an *event-triggered* version of equation 24. Here we assume that other nodes accesses a *sampled* version of the link state. In particular, let's associate a sequence of *sampling* instants, $\{T_j^L[\ell]\}_{\ell=0}^\infty$ with the jth node. The time $T_j^L[\ell]$ denotes the instant when the jth node samples its link state μ_j for the ℓth time and transmits that state to nodes $i \in \mathcal{S}_j$. We can see that at any time $t \in \Re$, the sampled link state is a piecewise constant function of time in which

$$\hat{\mu}_j(t) = \mu_j(T_j^L[\ell]) \quad (27)$$

for all $\ell = 0, \cdots, \infty$ and any $t \in [T_j^L[\ell], T_j^L[\ell+1])$. In this regard, the "event-triggered" version of equation 24 takes the form

$$x_i(t) = \int_0^t \left(\frac{\partial U_i(x_i(s))}{\partial x_i} - \sum_{j \in \mathcal{L}_i} \hat{\mu}_j(s) \right)^+_{x_i(s)} ds \quad (28)$$

for all ℓ and any $t \in [T_j^L[\ell], T_j^L[\ell+1])$.

The sequence $\{T_j^L[\ell]\}_{\ell=0}^\infty$ represents time instants when the node transmits its link state to the nodes that it relays traffic for. Under event-triggering, it will be convenient to have a similar flow of information from the node to the nodes that it uses as a relay node. We assume that node j can directly measure the total flow rate, $\sum_{i \in \mathcal{L}_j} x_i(t)$, in a continuous manner. The event-triggering scheme proposed below will require that node j have some knowledge of the time derivative of node $i \in \mathcal{L}_j$'s data rate. In particular, let $z_i(t)$ denote the time derivative of this data rate. $z_i(t)$ therefore satisfies

$$z_i(t) = \dot{x}_i(t) = \left(\nabla U_i(x_i(t)) - \sum_{j\in\mathcal{L}_i} \hat{\mu}_j(t)\right)^+_{x_i(t)} \quad (29)$$

for all $i \in \mathcal{V}$. We will refer to z_i as the ith node's *user state*. We associate a sequence $\{T_i^S[\ell]\}_{\ell=0}^\infty$ to each node $i \in \mathcal{V}$. The time $T_i^S[\ell]$ is the ℓth time when node i transmits its user state to all nodes $j \in \mathcal{L}_i$. We can therefore see that at any time $t \in \Re$, the sampled user state is a piecewise constant function of time satisfying

$$\hat{z}_i(t) = z_i(T_i^S[\ell]) \quad (30)$$

for all $\ell = 0, \cdots, \infty$ and any $t \in [T_i^S[\ell], T_i^S[\ell+1])$.

Next we will state the main theorem of this section. The proof will be found in the appendix.

THEOREM 5.1. *Consider the Lagrangian in equation 22 where the functions U_i are twice differentiable, strictly increasing, and strictly concave and where the routing matrix A is of full rank. Assume a fixed penalty parameter $w > 0$ and vector $\lambda = 0$. Consider the sequences $\{T_i^S[\ell]\}_{\ell=0}^\infty$ and $\{T_j^L[\ell]\}_{\ell=0}^\infty$ for each $i \in \mathcal{V}$, and each $j \in \mathcal{V}$, respectively. For each $i \in \mathcal{V}$, let the data rate, $x_i(t)$, satisfy equation 28 with sampled link states given by equation 27. For each $i \in \mathcal{V}$ let the user state $z_i(t)$ satisfy equation 29 and assume node j's measurement of the user state satisfies equation 30.*

Let ρ be a constant such that $0 < \rho < 1$. Assume that for all $i \in \mathcal{V}$ and all $\ell = 0, \cdots, \infty$, that

$$z_i^2(t) - \rho \hat{z}_i^2(t) \geq 0 \quad (31)$$

for $t \in [T_i^S[\ell], T_i^S[\ell+1])$. Further assume that for all $j \in \mathcal{V}$ and all $\ell = 0, \cdots, \infty$ that

$$\rho \sum_{i\in\mathcal{S}_j} \frac{1}{\overline{L}} \hat{z}_i^2(t) - \overline{LS}\left(\mu_j(t) - \hat{\mu}_j(t)\right)^2 \geq 0 \quad (32)$$

for $t \in [T_j^L[\ell], T_j^L[\ell+1])$. Then the data rates $x(t)$ asymptotically converge to the unique minimizer of $L(x; \lambda, w)$. ■

Theorem 5.1 provides the basis for constructing an event-triggered message-passing protocol. This theorem essentially asserts that we need to select the transmit times $\{T_i^S[\ell]\}$ and $\{T_j^L[\ell]\}$ so that the inequalities in equations 31 and 32 always hold. One obvious way to do this is to use the violation of these inequalities to trigger the sampling and transmission of link/user states across the network. At time $t = T_i^S[\ell]$, the inequality in equation 31 is automatically satisfied. After this sampling instant, $z_i(t)$ continues to change until the inequality is violated. We let that time instant be $T_i^S[\ell+1]$ and transmit the sampled user state to the nodes $j \in \mathcal{L}_i$. Simultaneously, node j compares the square of the error between the last transmitted link state $\hat{\mu}_j$ and the current link state μ_j. At the sampling time $T_j^L[\ell]$, this difference is zero and the inequality is trivially satisfied. After that time, $\mu_j(t)$ continues to change or the node may receive an updated user state $\hat{z}_i$ that may result in the violation of the inequality. We let that time be the next sampling instant, $T_j^L[\ell+1]$ and then transmit the sampled link state $\hat{\mu}_j$ to the nodes $i \in \mathcal{S}_j$. Note that each sensor node has two threads, one to adjust data rate and transmit user state if necessary, and the other to measure link state and transmit it if necessary.

The threshold conditions shown in equations 31-32 provide the basis for an event-triggered implementation of the algorithm presented earlier in section 4. Next we will present such an event-triggered distributed algorithm.

Future discussion needs an additional notation. For a function $f(t)$ defined on $t \in [0, T)$, denote $f^+(T)$ as the limit of $f(t)$ when t approaches T from the left hand side.

Each sensor node $i \in \mathcal{V}$ has an user algorithm and a link algorithm. It executes the following user algorithm. The main assumption here is that node i is continuously transmitting data at rate $x_i(t)$ at time t.

ALGORITHM 5.1. **Node i's User Update Algorithm**

1. **Parameter Initialization:** *Set the initial data rate $x_i^0 > 0$. Let $T = 0$.*

2. **State Initialization:** *Wait for nodes $j \in \mathcal{L}_i$ to send their link states $\mu_j(T)$ and set $\hat{\mu}_j = \mu_j(T)$. Initialize the user state to*

$$z_i(T) = \left(\nabla U_i(x_i^0) - \sum_{j\in\mathcal{L}_i} \hat{\mu}_j\right)^+_{x_i(T)} \quad (33)$$

set $\hat{z}_i = z_i(T)$ and transmit $z_i(T)$ to all nodes in $j \in \mathcal{L}_i$.

3. **Update Data Rate:** *Integrate the data rate equation*

$$x_i(t) = \int_T^t z_i(s)ds \quad (34)$$

$$z_i(t) = \left(\nabla U_i(x_i(t)) - \sum_{j\in\mathcal{L}_i} \hat{\mu}_j\right)^+_{x_i(t)} \quad (35)$$

$$x_i(T) = x_i^0 \quad (36)$$

where $t \in [T, T^+)$ and T^+ is the time instant when one of the following conditions is true

   (a) If $z_i^2(t) - \rho \hat{z}_i^2 \leq 0$ then broadcast $z_i^+(T^+)$ to all nodes $j \in \mathcal{L}_i$, and set $\hat{z}_i = z_i^+(T^+)$.

(b) *Or if node i receives a new link state $\mu_j^+(T^+)$ from node $j \in \mathcal{L}_i$, set $\hat{\mu}_j = \mu_j^+(T^+)$.*

4. **Increment Time:** *Set $T = T^+$, $x_i^0 = x_i^+(T^+)$ and go to step 3.*

Each sensor node $j \in \mathcal{V}$ also executes the following link algorithm. The main assumption here is that node j can continuously monitor its link state $\mu_j(t)$ at any time $t \in \Re$.

ALGORITHM 5.2. **Node j's Link Update Algorithm**

1. **Parameter Initialization:** *Set $T = 0$, $w_j > 0$.*
2. **State Initialization** *Measure the local link state*

$$\mu_j(T) = \max\left\{0, \frac{1}{w_j}\left(\sum_{i\in\mathcal{S}_j} x_i(T) - c_j\right)\right\} \quad (37)$$

Transmit $\mu_j(T)$ to all nodes $i \in \mathcal{S}_j$ and set $\hat{\mu}_j = \mu_j(T)$. Wait for nodes to return $z_i(T)$ for all $i \in \mathcal{S}_j$, and set $\hat{z}_i = z_i(T)$.

3. **Link Update:** *Continuously monitor the link state $\mu_j(t)$ for all $t \in [T, T^+)$ where T^+ is the time instant when one of the following events occur*

(a) *If*

$$\rho \sum_{i\in\mathcal{S}_j} \frac{1}{\overline{\overline{L}}}\hat{z}_i^2 \leq \overline{LS}\left(\mu_j(t) - \hat{\mu}_j\right)^2$$

then set $\hat{\mu}_j = \mu_j^+(T^+)$ and broadcast the updated link state $\mu_j^+(T^+)$ to all nodes $i \in \mathcal{S}_j$.

(b) *Or if node j receives a new user state $z_i^+(T^+)$ for any $i \in \mathcal{S}_j$, then set $\hat{z}_i = z_i^+(T^+)$.*

4. **Increment Time:** *Set $T = T^+$ and go to step 3.*

By theorem 5.1, the data rates $x(t)$ generated by algorithms 5.1- 5.2 converge asymptotically to the unique minimizer of $L(x; \lambda, w)$, which is an approximate solution to the NUM problem.

6. SIMULATION

This section presents simulation results. We assume tree communication structure in the data gathering problem. The sink node is the root node of the tree, and all leaf nodes and intermediate nodes send data to the root node. We compare the number of message exchanges of our event-triggered algorithm against the dual decomposition algorithm on the data gathering problem. Simulation results show that our event-triggered algorithm reduces the number of message exchanges by two order magnitude when compared to dual decomposition. Moreover, our algorithm enjoys better scalability with respect to the depth of the tree $\overline{D}$ and the maximum branch number of the tree $\overline{B}$. The remainder of this section is organized as follows: Subsection 6.1 discusses the simulation setup. The effect of different tradeoff parameter ρ is discussed in subsection 6.2. Simulation results on broadcast periods of our event-triggered algorithm are shown in subsection 6.3. The scalability results with respect to the maximum branch number of the tree and the depth of the tree are presented in subsection 6.4 and 6.5, respectively. Simulation results in subsection 6.2-6.5 talk about how fast our event-triggered algorithm converges to a small neighborhood of the equilibrium, and this is difficult to quantifiy if we use a time-varying constraint $Ax \leq c(t)$. For that reason, we use $c(t) = c$ in those simulations. Finally in subsection 6.6, we give an example where $c(t)$ is time varying, which shows our algorithm is able to track the variations in the energy constraint.

6.1 Simulation Setup

Denote $s \in \mathcal{U}[a, b]$ if s is a random variable uniformly distributed on $[a, b]$. Given $\overline{D}$ and $\overline{B}$, we randomly generate a tree with depth $\overline{D}$, and the number of branches each root/intermediate node in the tree has is in $\mathcal{U}[0, \overline{B}]$. We make sure that at least one node has $\overline{B}$ branches. After the network is generated, we assign utility function $U_i(x_i) = \alpha_i \log x_i$ for each node i, where $\alpha_i \in \mathcal{U}[0.8, 1.2]$. Each node i is assigned capacity $\overline{c}_i \in \mathcal{U}[32kbps, 48kbps]$. Each pair of transmission nodes are assumed to be $200m$ apart, and from equation 5, $e_{tx} = 4050nJ/bit$. Each node is has initial energy $E_i(0) = 100J$, and the expected lifetime $T_l = 1000s$. Once the network is generated, both algorithms are simulated. Remember $c(t)$ is defined in equation 7. For simulations in subsection 6.2-6.5, we use $c = c(0)$. The optimal rate x^* and its corresponding utility U^* are calculated using a global optimization technique.

Define error as (for both algorithms)

$$e(k) = \left|\frac{U(x(k)) - U^*}{U^*}\right| \quad (38)$$

where $x(k)$ is the rate at the kth iteration. $e(k)$ is the 'normalized deviation' from the optimal point at the kth iteration. In both algorithms, we count the number of iterations K for $e(k)$ to decrease to and stay in the neighborhood $\{e(k)|e(k) \leq e_d\}$. In dual decomposition, message passings occur at each iteration synchronously. So K is a measure of the total number of message exchanges. In our event-triggered algorithm, link events and user events occur in a totally asynchronous way. We add the total number of triggered events and divide this number by the number of sensor nodes N.

This works as an equivalent iteration number K for our event-triggered algorithm, and is a measure of the total number of message exchanges.

The default settings for simulation are as follows: $\rho = 0.5$, $e_d = 2\%$, $\overline{D} = 3$, $\overline{B} = 5$, $N = 25$. For both algorithms, the initial condition $x_i(0) \in \mathcal{U}[0.25kbps, 0.50kbps]$, $\forall i \in \mathcal{V}$. In dual decomposition, initial price $p_i = 0$ for $i \in \mathcal{V}$, and the step size γ is calculated using equation 19. In our event-triggered algorithm, $\lambda_i = 0$, $w_i = 0.01$ for $i \in \mathcal{V}$.

6.2 Effect of tradeoff coefficient ρ

In this subsection we discuss the effect of the tradeoff coefficient ρ on the performance of our event-triggered algorithm.

In our event-triggered algorithm, we only require the parameter ρ to be in the region $(0, 1)$. Recall ρ is a tradeoff between triggering the user event and link event. It is then natural to ask what impact ρ will have on the number of user events, link events and total events.

To see the effect of different ρ on the algorithm, we vary ρ from 0.01 to 0.99, while keeping all other parameters unchanged. The resulting figure 1 plots the event count (in logarithm scale) as a function of ρ. The dotted line at the bottom represents the total number of triggered link events. The dashed line in the middle is the total number of triggered user events, while the solid line on top corresponds to the total number of events.

We can see from figure 1 that the number of user events increases superlinearly with respect to ρ, while the number of link events is relatively insensitive to the changes in ρ. When ρ is small, the link events contribute to the major part in the total number of events, and when ρ is large, the user events contribute to the major part in the total number of events. The total number of events in this simulation increases with respect to ρ, and it suggests that we should choose small ρ in our event-triggered algorithm. Choosing $\rho \in [0.05, 0.5]$ will avoid have a large number of total events.

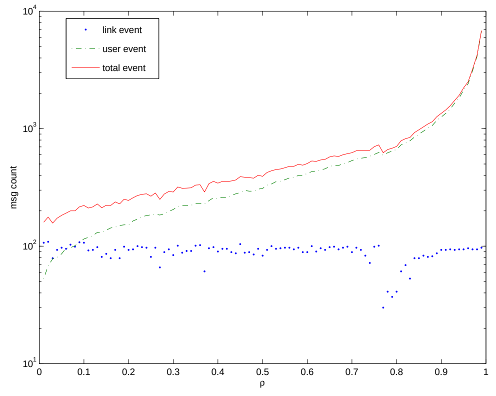

Figure 1: Event count as a function of ρ.

6.3 Broadcast periods of the event-triggered algorithm

In this subsection we present simulation results on the broadcast periods of our event-triggered algorithm. This simulation uses the default settings in subsection 6.1 and ran for 26.58s. For reference, with the same settings, the average broadcast period for the dual decomposition is 0.0352. In our event-triggered algorithm, there are a total of 83 link events, and 310 user events. So the sensor nodes have an average broadcast period of 1.6908, which is 48 times longer than in dual decomposition.

To see how the broadcast periods vary for each sensor node, we pick two nodes in the network, one is a congested node, and the other is an uncongested node. Their broadcast results are shown in figure 2. The two plots above correspond to the congested node, and the two plots below correspond to the uncongested node. The top left plot in figure 2 is the time history of broadcast periods generated by the node's local event (both user events and link events). The top right plot is the histogram of this node's broadcast periods. This node's broadcast periods range between 0.0200 and 2.8800. This node was triggered 69 times, with an average broadcast period of 0.3814. Although the minimum broadcast period is a little smaller than dual decomposition, but the average period is 11 times longer than in dual decomposition. For the uncongested node, its broadcast periods range between 0.0600 and 4.5200. This node was triggered 16 times, with an average broadcast period of 1.6462. The uncongested node enjoys much longer average broadcast period.

As we can see from figure 2, for the uncongested nodes, the number of triggered events are relatively small, and they have long average broadcast periods. For the congested nodes, the number of triggered events are large, which results in short average broadcast periods. The average broadcast period in our event-triggered algorithm is much longer than in dual decomposition. The communication strategy in our event-triggered algorithm is more adaptive, and the broadcast periods are adjusted based on the current situation of the network.

6.4 Scalability with respect to $\overline{B}$

In this simulation, we fix the depth of the tree $\overline{D} = 3$, and vary $\overline{B}$ from 2 to 10. For each $\overline{B}$, both algorithms were run 500 times, and each time a random network which satisfies the above specification is generated. The mean m_K and standard deviation σ_K of K are computed for each $\overline{B}$. m_k works as our criteria for comparing the scalability of the two algorithms. Figure 3

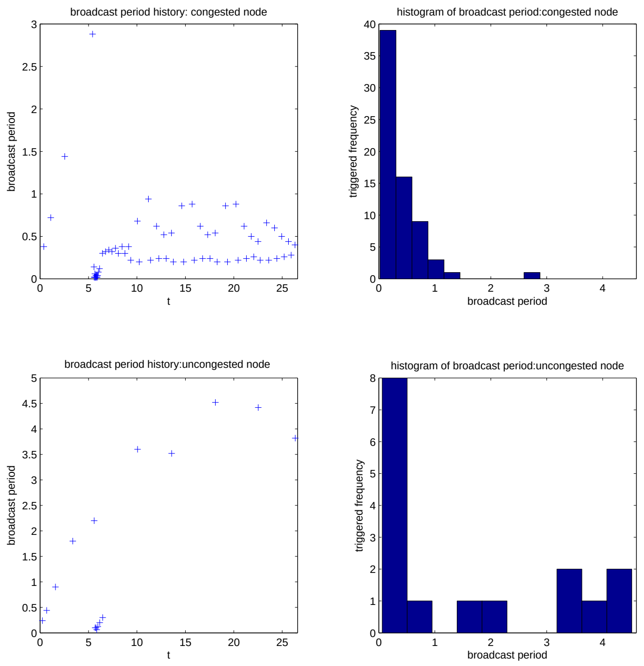

Figure 2: Broadcast results for two nodes

plots the iteration number K (in logarithm scale) as a function of $\overline{B}$ for both algorithms. The asterisks above represent m_K for dual decomposition, while the circles below correspond to our event-triggered algorithm. The dotted vertical line around each asterisk and circle corresponds to the interval $[m_K - \sigma_K, m_K + \sigma_K]$ for each different $\overline{B}$ denoted by the x-axis.

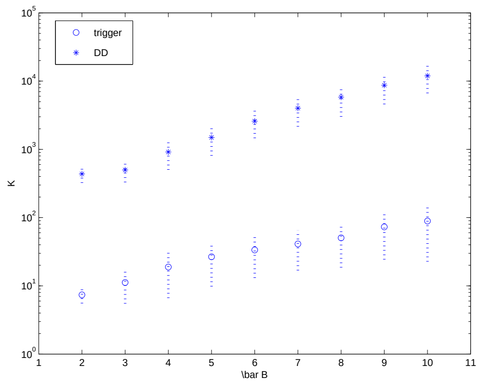

Figure 3: Iteration number K as a function of $\overline{B}$ for both algorithms.

For our event-triggered algorithm, when $\overline{B}$ increases from 2 to 10, m_K increases from 8 to 89. σ_K at the same time increases from 2 to 66. For dual decomposition, m_K increases from 0.0437×10^4 to 1.1908×10^4. σ_K at the same time increases from 0.1137×10^3 to 5.2532×10^3. Our event-triggered algorithm is about two order magnitude faster than the dual decomposition. We can also see that, our event-triggered algorithm enjoys better scalability. As $\overline{B}$ increases from 2 to 10, m_K increases by 10 times for event-triggered algorithm, while for dual decomposition, m_K increases by 27 times.

6.5 Scalability with respect to $\overline{D}$

This simulation is similar to subsection 6.4 except that we fix $\overline{B} = 5$, and vary $\overline{D}$ from 2 to 5. Figure 4 plots K (in logarithm scale) as a function of $\overline{D}$ for both algorithms. For our event-triggered algorithm, when $\overline{D}$ increases from 2 to 5, m_K increases from 24 to 125. σ_K at the same time varies between 12 and 70. For dual decomposition, m_K increases from 0.0412×10^4 to 2×10^4. σ_K at the same time increases from 0.076×10^3 to 6.495×10^3. We should point out that, since dual decomposition scales poorly with $\overline{D}$, sometimes it takes tremendously long time to converge. In the simulation, if the e_d neighborhood is not reached after 2×10^4 iterations, we simply count $K = 2 \times 10^4$.

Our event-triggered algorithm is about two order magnitude faster than the dual decomposition. We can also see that, our event-triggered algorithm enjoys better scalability. As $\overline{D}$ increases from 2 to 5, m_K increases by 5 times for event-triggered algorithm, while for dual decomposition, m_K increases by 48 times.

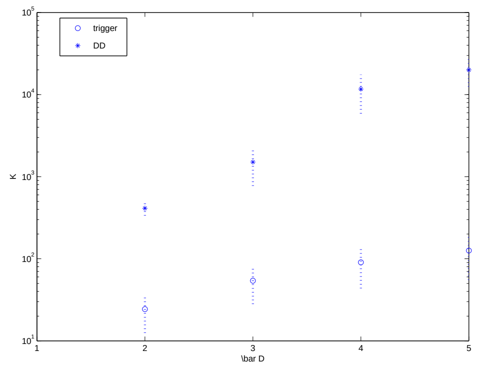

Figure 4: Iteration number K as a function of $\overline{D}$ for both algorithms.

6.6 A time-varying example

In this simulation, $c(t)$ varies according to equation 7. We want to see how well our event-triggered algorithm tracks the changes in $c(t)$. We run both algorithms until one of the sensor nodes in the network has only $10J$ energy left. Figure 5 plots the total utility $U(x(k))$ obtained as a function of the equivalent iteration number k (in logarithm scale) for both algorithms. As we can see from the figure, both algorithms track the changes in $c(t)$. Instead of converging to the original optimal utility at 85, both trajectories converge to the current optimal utility at around 65. It is also easy to see from figure 5 that our event-triggered algorithm is about two order magnitude faster than dual decomposition.

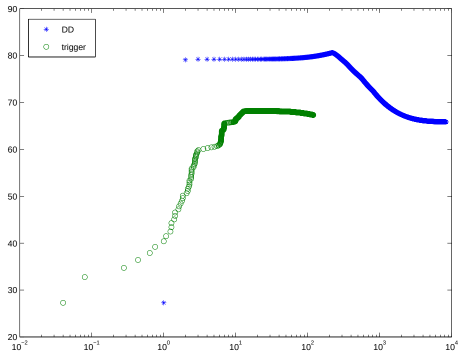

Figure 5: $U(x(k))$ as a function of equivalent iteration number k for both algorithms.

7. CONCLUSION

This paper introduces the use of event-triggered distributed algorithm to solve the optimization problems in sensor networks. The event-triggered algorithm can greatly reduce the message passing complexity in distributed algorithms. We give a general class of optimization problems in sensor networks where the event-triggered algorithm can be used. We then use the data gathering problem as an example, and propose an event-triggered distributed algorithm for the data gathering problem and prove its convergence. Simulation results show that the proposed algorithm reduces the number of message exchanges by two orders of magnitude compared to commonly used dual decomposition algorithms. It also enjoys better scalability. Future work will use the event-triggered algorithm to solve other problems mentioned in the paper.

8. APPENDIX

8.1 Proof of Theorem 5.1

Proof. For convenience, we do not explicitly include the time dependence of $x_i(t)$, $\hat{x}_i(t)$, $z_i(t)$, $\hat{z}_i(t)$, $\mu_j(t)$, $\hat{\mu}_j(t)$ in most part of the proof. For all $t \geq 0$ we have

$$
\begin{aligned}
-\dot{L}(x;\lambda,w) &= -\sum_{i=1}^N \frac{\partial L}{\partial x_i}\frac{dx_i}{dt} \\
&= \sum_{i=1}^N z_i[\nabla U_i(x_i) - \sum_{j=1}^N \mu_j A_{ji}] \qquad (39)\\
&\geq \sum_{i=1}^N \left\{ \frac{1}{2}z_i^2 - \frac{1}{2}[\sum_{j=1}^N (\mu_j - \hat{\mu}_j)A_{ji}]^2 \right\} \qquad (40)
\end{aligned}
$$

The last inequality holds whether the positive projection is active or not for each node i. Also remember there are only $|\mathcal{L}_i|$ nonzero terms in the sum $\sum_{j=1}^N(\mu_j - \hat{\mu}_j)A_{ji}$, then by using the inequality

$$
-\left[\sum_{j=1}^N (\mu_j - \hat{\mu}_j)A_{ji}\right]^2 \geq -|\mathcal{L}_i|\sum_{j=1}^N [(\mu_j - \hat{\mu}_j)A_{ji}]^2 \tag{41}
$$

we have $-\dot{L}(x;\lambda,w)$

$$
\begin{aligned}
&\geq \frac{1}{2}\sum_{i=1}^N z_i^2 - \frac{1}{2}\sum_{i=1}^N \left\{ |\mathcal{L}_i| \sum_{j=1}^N [(\mu_j - \hat{\mu}_j)A_{ji}]^2 \right\} \qquad (42)\\
&= \frac{1}{2}\sum_{i=1}^N z_i^2 - \frac{1}{2}\sum_{j=1}^N \left\{ (\mu_j - \hat{\mu}_j)^2 \sum_{i=1}^N |\mathcal{L}_i| A_{ji}^2 \right\} \qquad (43)\\
&\geq \frac{1}{2}\sum_{i=1}^N z_i^2 - \frac{1}{2}\sum_{j=1}^N \overline{LS}(\mu_j - \hat{\mu}_j)^2 \qquad (44)
\end{aligned}
$$

Consider the term $\frac{1}{2}\rho\sum_{i=1}^N \hat{z}_i^2$, we have

$$
\begin{aligned}
\frac{1}{2}\rho\sum_{i=1}^N \hat{z}_i^2 &= \frac{1}{2}\rho\sum_{i=1}^N \overline{L}\frac{1}{\overline{L}}\hat{z}_i^2 \qquad (45)\\
&= \frac{1}{2}\rho\sum_{j=1}^N\sum_{i=1}^N \frac{1}{\overline{L}}\hat{z}_i^2 A_{ji} + \frac{1}{2}\rho\sum_{i=1}^N(\overline{L} - |\mathcal{L}_i|)\frac{1}{\overline{L}}\hat{z}_i^2 \qquad (46)
\end{aligned}
$$

Remember $|\mathcal{L}_i| \leq \overline{L}$ for $i \in \mathcal{V}$, this means

$$
\begin{aligned}
-\dot{L}(x;\lambda,w) \geq\ & \frac{1}{2}\sum_{i=1}^N z_i^2 - \frac{1}{2}\rho\sum_{i=1}^N \hat{z}_i^2 \\
& + \frac{1}{2}\rho\sum_{i=1}^N \hat{z}_i^2 - \frac{1}{2}\sum_{j=1}^N \overline{LS}(\mu_j - \hat{\mu}_j)^2 \qquad (47)\\
\geq\ & \frac{1}{2}\sum_{i=1}^N [z_i^2 - \rho\hat{z}_i^2] + \\
& \frac{1}{2}\sum_{j=1}^N \left\{ \rho\sum_{i\in\mathcal{S}_j}\frac{1}{\overline{L}}\hat{z}_i^2 - \overline{LS}(\mu_j - \hat{\mu}_j)^2 \right\} \qquad (48)
\end{aligned}
$$

which immediately suggests us if the sequences of sampling instants $\{T_i^S[\ell]\}_{\ell=0}^\infty$ and $\{T_j^L[\ell]\}_{\ell=0}^\infty$ satisfy the inequalities in equation 31 and 32 for all $\ell = 0, 1, 2, ..., \infty$, and any $i \in \mathcal{V}$, $j \in \mathcal{V}$, then $\dot{L}(x;\lambda,w) \leq 0$ is guaranteed for all t.

By using the properties of $U_i(x_i)$ and $\psi_j(x;\lambda,w)$, it is easy to show that for any fixed λ and w, $L(x;\lambda,w)$ is strictly convex in x. It thus has a unique minimizer. Suppose $x^*(\lambda,w)$ is this minimizer, and the corresponding Lagrangian is $L(x^*;\lambda,w)$. Define $V(x) = L(x;\lambda,w) - L(x^*;\lambda,w)$. It is trivial to see $V(x)$ is a Lyapunov function for the system. Moreover, $\dot{V}(x) = 0$ means $\dot{L}(x;\lambda,w) = 0$. The only scenario this can happen is

$$z_i = \hat{z}_i = 0, \quad \forall i \in \mathcal{V}, \quad \mu_j = \hat{\mu}_j, \quad \forall j \in \mathcal{V} \qquad (49)$$

which corresponds to $x^*(\lambda,w)$. As a result, the equilibrium $x^*(\lambda,w)$ is asymptotically stable. Proof complete. □

9. REFERENCES

[1] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," *Proceedings of the third international symposium on Information processing in sensor networks*, pp. 20–27, 2004.

[2] A. Speranzon, C. Fischione, and K. Johansson, "Distributed and Collaborative Estimation over Wireless Sensor Networks," *Proceedings of the IEEE Conference on Decision and Control*, pp. 1025–1030, 2006.

[3] S. Samar, S. Boyd, and D. Gorinevsky, "Distributed estimation via dual decomposition," *European Control Conference*, 2007.

[4] B. Johansson, M. Rabi, and M. Johansson, "A simple peer-to-peer algorithm for distributed optimization in sensor networks," *IEEE Conference on Decision and Control*, 2007.

[5] W. Chen and L. Sha, "An energy-aware data-centric generic utility based approach in wireless sensor networks," *IPSN*, pp. 215–224, 2004.

[6] W. Chen, J. Hou, L. Sha, and M. Caccamo, " A distributed, energy-aware, utility-based approach for data transport in wireless sensor networks," *Proceedings of the IEEE Milcom*, 2005.

[7] R. Madan and S. Lall, "Distributed algorithms for maximum lifetime routing in wireless sensor networks," in *IEEE GLOBECOM'04*, vol. 2.

[8] P. Wan and M. Lemmon, "Distributed Flow Control using Embedded Sensor-Actuator Networks for the Reduction of Combined Sewer Overflow (CSO) Events," *IEEE Conference on Decision and Control*, 2007.

[9] Y. Qiu and P. Marbach, "Bandwidth allocation in ad hoc networks: a price-based approach," *INFOCOM 2003*.

[10] Y. Xue, B. Li, and K. Nahrstedt, "Optimal resource allocation in wireless ad hoc networks: a price-based approach," *IEEE Transactions on Mobile Computing*, vol. 5, no. 4, pp. 347–364, 2006.

[11] L. Chen, S. Low, and J. Doyle, "Joint congestion control and media access control design for ad hoc wireless networks," *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 3, 2005.

[12] Y. Yi and S. Shakkottai, "Hop-by-hop congestion control over a wireless multi-hop network," *IEEE/ACM Transactions on Networking*, vol. 15, no. 1, pp. 133–144, 2007.

[13] S. Low and D. Lapsley, "Optimization flow control, I: basic algorithm and convergence," *IEEE/ACM Transactions on Networking (TON)*, vol. 7, no. 6, pp. 861–874, 1999.

[14] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient Communication Protocol for Wireless Microsensor Networks," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, vol. 2, 2000.

[15] F. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, 1998.

[16] N. Sadagopan and B. Krishnamachari, "Maximizing data extraction in energy-limited sensor networks," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 3, 2004.

[17] L. Montestruque and M. Lemmon, "Csonet: a metropolitan scale wireless sensor-actuator network," *International Workshop on Mobile Device and Urban Sensing (MODUS)*, 2008.

[18] J. Wen and M. Arcak, "A unifying passivity framework for network flow control," *Automatic Control, IEEE Transactions on*, vol. 49, no. 2, pp. 162–174, 2004.

[19] D. Palomar and M. Chiang, "Alternative Distributed Algorithms for Network Utility Maximization: Framework and Applications," *Automatic Control, IEEE Transactions on*, vol. 52, no. 12, pp. 2254–2269, 2007.

[20] D. Bertsekas, *Nonlinear programming.* Athena Scientific, 1999.

[21] P. Wan and M. D. Lemmon, "Distributed Network Utility Maximization using Event-triggered augmented Lagrangian methods," *Submitted to American Control Conference 2009.*