An event-triggered distributed primal-dual algorithm for Network Utility Maximization

Pu Wan and Michael D. Lemmon

***Abstract*— Many problems associated with networked systems can be formulated as network utility maximization (NUM) problems. NUM problems maximize a global separable measure of network optimality subject to linear constraints on resources. Dual decomposition is a widely used distributed algorithm that solves the NUM problem. This approach, however, uses a step size that is inversely proportional to measures of network size such as maximum path length or maximum neighborhood size. As a result, the number of messages exchanged between nodes by a dual decomposition scales poorly with respect to these measures. This paper presents a distributed primal-dual algorithm for the NUM problem that uses event-triggering. Under event triggering, each agent broadcasts to its neighbors when a local "error" signal exceeds a state dependent threshold. The paper establishes such state-dependent event-triggering thresholds under which the proposed algorithm converges. The paper gives an upper bound on the largest number of successive data dropouts the network can tolerate while ensuring the algorithm's convergence. State-dependent lower bound on the broadcast period is also given. Simulation results show that the proposed algorithm reduce the number of message exchanges by up to two orders of magnitude, and is scale-free with respect to the above two measures of network size.**

I. INTRODUCTION

A networked system is a collection of subsystems where individual subsystems exchange information over some communication network. Examples of networked systems include the electrical power grid, wireless sensor networks, and the Internet. In many of these networked systems, we're interested in optimizing overall system behavior subject to local constraints generated by limited subsystem resources. Examples of such applications include parameter estimation in sensor networks [1] [2], distributed control of sensor-actuator networks [3], resource allocation in wireless communication networks [4] [5] [6] [7], and congestion control in wired communication networks [8] [9]. In all of these problems, we find subsystems communicating with each other in order to collaboratively solve a network optimization problem.

[4] [5] [6] [7] [8] [9] fall into the general framework of Network Utility Maximization (NUM) problems. NUM problems maximize a global separable measure of the networked system's performance subject to linear constraints on resources. A variety of distributed algorithms [9] [10] [11] have been proposed to solve the NUM problem after Kelly's seminar work [8]. These algorithms can be classified as either primal, dual, or primal-dual algorithms, depending upon whether the user, the link, or both user and link, respectively, update their states through gradient following recursions. Among all existing algorithms, the dual decomposition approach proposed by Low et al. [9] is the most widely used algorithm for the NUM problem. Low et al. showed that their dual decomposition algorithm was stable for a step size that is inversely proportional to two important measures of network size: the maximum length path $\overline{L}$, and the maximum number of neighbors $\overline{S}$. So as these two measures get large, the step size required for stability becomes extremely small. Step size, of course, determines the number of computations required for the algorithm's convergence. Under dual decomposition, system agents exchange information at each iteration, so that step size also determines the message passing complexity of the dual decomposition algorithm. Therefore if we use the "stabilizing" step size, dual decomposition will have a message complexity that scales in a super-linear manner with those two measures of network size, $\overline{L}$ and $\overline{S}$.

For many networked systems this type of message passing complexity may be unacceptable. This is particularly true for systems communicating over a wireless network. In such networked systems, the energy required for communication can be significantly greater than the energy required to perform computation [12]. 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 of distributed NUM algorithms. The paper presents a distributed primal-dual NUM algorithm that uses event-triggered message passing. We prove that the proposed algorithm converges to an approximate solution of the NUM problem. We also consider the scenarios when there are data dropouts, and give an upper bound on the largest number of successive data dropouts the network can tolerate, while ensuring the asymptotic convergence of the algorithm. State-dependent lower bound on the broadcast period is also given. Simulation results show that the algorithm has a message passing complexity that is up to two orders of magnitude lower than dual decomposition, and is scale-free with respect to the above two measures of network size.

This work is similar to our prior work in [13]. Both work are based on the augmented Lagrangian method for the NUM problem. However, in [13], we eliminate the dual variables from the Lagrangian function explicitly, which as a result gives us a primal algorithm. In the primal algorithm

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).

in [13], there is an event associated with each user and link. Each user has to know the gradient information of its own data rate, which is very undesirable. Moreover, the interactions between the user events and link events complicate the algorithm significantly, and make the analysis of the algorithm very difficult. In the primal-dual algorithm in this paper, there are only link events. The algorithm has comparable performance to the primal algorithm, but the simplicity in the event structure enables us to obtain some additional analytical results. In [13], the primal algorithm converges to the exact minimizer of the NUM problem. To be specific, we included a distributed update strategy for the penalty parameters w. So as w goes to zero, the exact minimizer is reached. We could develop a similar distributed update strategy so that the primal-dual algorithm also converges to the exact minimizer. However, for the purpose of this paper, we only considering the problem of converging to an approximate minimizer.

The rest of the paper is organized as follows. Section II formally states the NUM problem and reviews the dual decomposition algorithm. The event-triggered distributed primal-dual algorithm is based on a basic primal-dual algorithm for the NUM problem, which will be first discussed in section III. Section IV then presents the event-triggered distributed primal-dual algorithm, and proves its convergence. Section V and VI analyzes data dropouts and broadcast period in the event-triggered primal-dual algorithm, respectively. Finally, section VII presents simulation results and section VIII concludes the paper.

II. Dual Decomposition NUM Algorithm

NUM problem [8] considers a network of N users and M links. We let $\mathcal{S} = \{1, \cdots, N\}$ denote the set of users and $\mathcal{L} = \{1, \cdots, M\}$ denote the set of links. Each user generates a flow with a specified data rate. Each flow may traverse several links (which together constitute a route) before reaching its destination. The set of links that are used by user $i \in \mathcal{S}$ will be denoted as $\mathcal{L}_i$ and the set of users that are using link $j \in \mathcal{L}$ will be denoted as $\mathcal{S}_j$. The NUM problem takes the form

$$\begin{array}{ll} \text{maximize:} & U(x) = \sum_{i\in\mathcal{S}} U_i(x_i) \\ \text{subject to:} & Ax \le c \quad x \ge 0 \end{array} \tag{1}$$

where $x = [x_1, ..., x_N]^T$ and $x_i \in \mathbb{R}$ is user i's data rate. $A \in \mathbb{R}^{M\times N}$ is the routing matrix mapping users onto links and $c \in \mathbb{R}^M$ is a vector of link capacities. The ji'th component, A_{ji}, is 1 if user i's flow traverses link j and is zero otherwise. The jth row of Ax represents the total data rates going through link j, which cannot exceed its capacity c_j. The cost function U is the sum of the user *utility* functions $U_i(x_i)$. These utility functions represent the reward (i.e. quality-of-service) [8][9] user i gets by transmitting at rate x_i.

NUM problems are often solved using the dual decomposition algorithm [9]. The algorithm examines the dual of the NUM problem, which is

$$\begin{array}{ll} \text{minimize:} & \max_{x\ge 0} \left\{\sum_{i\in\mathcal{S}} U_i(x_i) - p^T(Ax - c)\right\} \\ \text{subject to:} & p \ge 0 \end{array} \tag{2}$$

where $p = \begin{bmatrix} p_1 & \cdots & p_M \end{bmatrix}^T$ is the Lagrange multiplier vector (which can be viewed as the price for using each link [8]) associated with the inequality constraint $Ax \le c$. 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. [9] established conditions under which a pair of recursions would generate a sequence of user rates, $\{x[k]\}_{k=0}^{\infty}$, and link prices, $\{p[k]\}_{k=0}^{\infty}$, that asymptotically converge to a solution of the dual problem. Given initial $x[0]$ and $p[0]$, then for all $i \in \mathcal{S}$ and $j \in \mathcal{L}$, we let

$$x_i[k+1] = \arg\max_{x_i\ge 0} \left\{ U_i(x_i[k]) - x_i[k] \sum_{j\in\mathcal{L}_i} p_j[k]) \right\} \tag{3}$$

$$p_j[k+1] = \max\left\{0, p_j[k] + \gamma\left\{\sum_{i\in\mathcal{S}_j} x_i[k] - c_j\right\}\right\} \tag{4}$$

for $k = 0, \cdots, \infty$. It is shown that a stabilizing step size is

$$0 < \gamma < \gamma^* = \frac{-2\max_{(i,x_i)} \nabla^2 U_i(x_i)}{\overline{L}\overline{S}} \tag{5}$$

where $\overline{L}$ is the maximum number of links any user uses and $\overline{S}$ is the maximum number of users any link has. Equation 5 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 $\overline{S}$.

III. Basic Primal-dual Algorithm

The event-triggered distributed primal-dual algorithm in this paper is also 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.

To apply the augmented Lagrangian method on our NUM problem, we need to introduce the slack variable $s \in \mathbb{R}^M$ and replace the inequalities $c_j - a_j^T x \ge 0$, $\forall j \in \mathcal{L}$ by

$$a_j^T x - c_j + s_j = 0, \quad s_j \ge 0, \quad \forall j \in \mathcal{L} \tag{6}$$

The *augmented cost* is then

$$L(x, s; \lambda, w) = -\sum_{i\in\mathcal{S}} U_i(x_i) + \sum_{j\in\mathcal{L}} \lambda_j (a_j^T x - c_j + s_j) + \frac{1}{2}\sum_{j\in\mathcal{L}} \frac{1}{w_j}(a_j^T x - c_j + s_j)^2 \tag{7}$$

Here a penalty parameter w_j is associated with each link constraint, and $w = [w_1, \cdots, w_M]$. Suppose λ_j^* is the Lagrange multiplier associated with constraint $c_j - a_j^T x \ge 0$ in the Karush-Kuhn-Tucker conditions of the NUM problem. λ_j is an estimate of λ_j^* and $\lambda = [\lambda_1, \cdots, \lambda_M]$. $a_j^T = [A_{j1}, \cdots, A_{jN}]$ is the jth row of the routing matrix A.

In our earlier work [13], we eliminate the dual variable s, and rewrite the augmented cost in equation 7 as a function of only the primal variable x for fixed λ and w. That approach gives us the resulting primal algorithm. In this paper, we treat $L(x, s; \lambda, w)$ as a function of both the primal variable x and dual variable s. This gives us a primal-dual algorithm.

In this primal-dual setup, the augmented Lagrangian method solves the NUM problem by approximately minimizing $L(x, s; \lambda[k], \overline{w}[k])$ for sequences of $\{\overline{w}[k]\}_{k=0}^{\infty}$ and $\{\lambda[k]\}_{k=0}^{\infty}$. Let $(x^*[k], s^*[k])$ denote the approximate minimizer for $L(x, s; \lambda[k], \overline{w}[k])$. Then for appropriately chosen sequences $\{\overline{w}[k]\}_{k=0}^{\infty}$ and $\{\lambda[k]\}_{k=0}^{\infty}$, the sequence of approximate minimizers $\{(x^*[k], s^*[k])\}_{k=0}^{\infty}$ converges to the optimal point of the NUM problem. The choices are as follows. $\{\overline{w}_j[k]\}_{k=0}^{\infty}$ are monotone decreasing to zero. $\{\lambda_j[k]\}_{k=0}^{\infty}$ satisfy $\lambda_j[k+1] = \max\{0, \lambda_j[k] + \frac{1}{\overline{w}_j[k]}(a_j^T x^*[k] - c_j + s_j^*[k])\}$.

In [13], we gave a primal algorithm based on the augmented Lagrangian method that converges to the exact minimizer of the NUM problem. However, in many scenarios, it usually suffices to obtain an approximate minimizer. So instead of minimizing $L(x, s; \lambda[k], \overline{w}[k])$ for sequences of $\{\overline{w}[k]\}_{k=0}^{\infty}$ and $\{\lambda[k]\}_{k=0}^{\infty}$, we are only considering the problem of minimizing $L(x, s; \lambda, w)$ for fixed λ and w in this paper. If $\lambda_j = 0$ and w_j is sufficiently small, the minimizer of $L(x, s; \lambda, w)$ will be a good approximation to the solution of the original NUM problem.

The **basic primal-dual algorithm** is given as follows:

1) **Initialization:** Select initial rate $x^0 > 0$, initial $s^0 \geq 0$. Set $\lambda_j = 0$ and sufficiently small $w_j > 0$, $j \in \mathcal{L}$.
2) **Recursive Loop: Minimize** $L(x, s; \lambda, w)$

$$\begin{aligned} x &= \max\left\{0, x^0 - \gamma \nabla_x L(x^0, s^0; \lambda, w)\right\} \quad (8) \\ s &= \max\left\{0, s^0 - \gamma \nabla_s L(x^0, s^0; \lambda, w)\right\} \quad (9) \\ (x^0, s^0) &= (x, s) \end{aligned}$$

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 $L(x, s; \lambda, w)$ using a simple gradient following method. γ is a sufficiently small step size.

The computations above can be easily distributed among the users and links. We will see that in our event-triggered distributed implementation of the algorithm in section IV.

In dual decomposition and the algorithm shown above, the exchange of information between users and links happens each time the gradient following update is applied. This means that the number of messages passed between users and links 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 following section presents an event-triggered distributed implementation of the basic primal-dual algorithm presented in this section.

IV. Event-triggered Distributed Primal-dual Algorithm

Implementing the primal-dual algorithm in section III in a distributed manner requires communication between users and links. An event-triggered implementation of the algorithm assumes that the transmission of messages between users and links 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 event-triggered distributed primal-dual algorithm.

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

$$\begin{aligned} x_i(t) &= -\int_0^t \left(\nabla_{x_i} L(x(\tau), s(\tau); \lambda, w)\right)_{x_i(\tau)}^+ d\tau \\ &= \int_0^t \left(\frac{\partial U_i(x_i(\tau))}{\partial x_i} - \sum_{j \in \mathcal{L}_i} \mu_j(\tau)\right)_{x_i(\tau)}^+ d\tau \quad (10) \end{aligned}$$

for each user $i \in \mathcal{S}$ and

$$\begin{aligned} s_j(t) &= -\int_0^t \left(\nabla_{s_j} L(x(\tau), s(\tau); \lambda, w)\right)_{s_j(\tau)}^+ d\tau \\ &= \int_0^t \left(-\mu_j(\tau)\right)_{s_j(\tau)}^+ d\tau \quad (11) \end{aligned}$$

for each link $j \in \mathcal{L}$, where

$$\mu_j(t) = \lambda_j + \frac{1}{w_j}\left(\sum_{i \in \mathcal{S}_j} x_i(t) - c_j + s_j(t)\right) \quad (12)$$

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

$$(f(x))_x^+ = \begin{cases} 0, & \text{if} \quad x = 0 \quad \text{and} \quad f(x) < 0 \\ f(x), & \text{otherwise} \end{cases} \quad (13)$$

The positive projection used in equation 10 and 11 guarantees the user rate $x_i(t)$ and *dual state* $s_j(t)$ are always nonnegative along the trajectory.

Equations 10 and 11 are the continuous-time versions of the update in equations 8 and 9, respectively. In equation 10, user $i \in \mathcal{S}$ can compute its rate only based on the information from itself, and the information of μ_j from those links that are being used by user i. Link $j \in \mathcal{L}$ is associated with a dynamical system which is characterized by equations 11-12. This first-order dynamical system takes the total flow rate that goes through link j as the input, and outputs μ_j. To make our notations consistent with [13], we call μ_j as the jth link's *local state*, which serves as the feedback signal to the users in $i \in \mathcal{S}_j$. From equations 11 and 12, link j only needs to be able to measure the total flow that goes through itself in order to compute its local state μ_j. All of this information is locally available, so both the user rate update and the link state computation can be done in a distributed manner.

In equation 10, the link state information is available to the users in a continuous manner. We now consider an *event-triggered* version of equation 10, where the user accesses a sampled version of the link state. In particular, we associate a sequence of *sampling* instants, $\{T_j^L[\ell]\}_{\ell=0}^{\infty}$ with the jth link. The time $T_j^L[\ell]$ denotes the instant when the jth link samples its link state μ_j in equation 12 for the ℓth time and transmits that state to users $i \in \mathcal{S}_j$. At any time $t \in \Re$, the sampled link state is a piecewise constant function of time which satisfies

$$\hat{\mu}_j(t) = \mu_j(T_j^L[\ell]) \tag{14}$$

for all $\ell = 0, \cdots, \infty$ and any $t \in [T_j^L[\ell], T_j^L[\ell+1])$. The event-triggered version of equation 10 takes the form

$$x_i(t) = \int_0^t \left(\frac{\partial U_i(x_i(\tau))}{\partial x_i} - \sum_{j \in \mathcal{L}_i} \hat{\mu}_j(\tau) \right)_{x_i(\tau)}^{+} d\tau \tag{15}$$

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

Next we will state the main theorem of this section.

Theorem 4.1: Consider the Lagrangian in equation 7 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 $\lambda \geq 0$. For each link $j \in \mathcal{L}$, consider the sequence $\{T_j^L[\ell]\}_{\ell=0}^{\infty}$, and its dynamics satisfy equations 11-12. For each user $i \in \mathcal{S}$, let the user rate, $x_i(t)$, satisfy equation 15 with sampled link states defined in equation 14.

For all $j \in \mathcal{L}$, let ρ_j be a constant such that $0 < \rho_j \leq 1$. Assume that for all $j \in \mathcal{L}$, and all $\ell = 0, \cdots, \infty$ that

$$\rho_j \left[(-\mu_j(t))_{s_j(t)}^{+} \right]^2 - \frac{1}{2}\overline{LS} \left[\mu_j(t) - \hat{\mu}_j(t) \right]^2 \geq 0 \tag{16}$$

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

Proof: Let $z_i(t)$ denote the time derivative of user i's flow rate, $z_i(t)$ therefore satisfies

$$z_i(t) = \dot{x}_i(t) = \left(\nabla U_i(x_i(t)) - \sum_{j \in \mathcal{L}_i} \hat{\mu}_j(t) \right)_{x_i(t)}^{+} \tag{17}$$

for all $i \in \mathcal{S}$. 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

$$-\dot{L}(x, s; \lambda, w) = -\sum_{i=1}^{N} \frac{\partial L}{\partial x_i}\frac{dx_i}{dt} - \sum_{j=1}^{M} \frac{\partial L}{\partial s_j}\frac{ds_j}{dt}$$

$$= \sum_{i=1}^{N} z_i \left[\nabla U_i(x_i) - \sum_{j=1}^{M} \mu_j A_{ji} \right] + \sum_{j=1}^{M} (-\mu_j)_{s_j}^{+}(-\mu_j)$$

$$\geq \sum_{i=1}^{N} \left\{ \frac{1}{2} z_i^2 - \frac{1}{2}[\sum_{j=1}^{M} (\mu_j - \hat{\mu}_j) A_{ji}]^2 \right\} + \sum_{j=1}^{M} \left[(-\mu_j)_{s_j}^{+} \right]^2$$

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

$$-\left[\sum_{j=1}^{M} (\mu_j - \hat{\mu}_j) A_{ji} \right]^2 \geq -|\mathcal{L}_i| \sum_{j=1}^{M} \left[(\mu_j - \hat{\mu}_j) A_{ji} \right]^2 \tag{18}$$

we have

$$\begin{aligned}
& -\dot{L}(x, s; \lambda, w) \\
\geq\ & \frac{1}{2}\sum_{i=1}^{N} z_i^2 - \frac{1}{2}\sum_{i=1}^{N} \left\{ |\mathcal{L}_i| \sum_{j=1}^{M} [(\mu_j - \hat{\mu}_j) A_{ji}]^2 \right\} \\
& + \sum_{j=1}^{M} \left[(-\mu_j)_{s_j}^{+} \right]^2 \\
=\ & \frac{1}{2}\sum_{i=1}^{N} z_i^2 - \frac{1}{2}\sum_{j=1}^{M} \left\{ (\mu_j - \hat{\mu}_j)^2 \sum_{i=1}^{N} |\mathcal{L}_i| A_{ji}^2 \right\} \\
& + \sum_{j=1}^{M} \left[(-\mu_j)_{s_j}^{+} \right]^2 \\
\geq\ & \frac{1}{2}\sum_{i=1}^{N} z_i^2 - \frac{1}{2}\sum_{j=1}^{M} \overline{LS} (\mu_j - \hat{\mu}_j)^2 + \sum_{j=1}^{M} \left[(-\mu_j)_{s_j}^{+} \right]^2 \\
\geq\ & \frac{1}{2}\sum_{i=1}^{N} z_i^2 + \sum_{j=1}^{M} (1 - \rho_j) \left[(-\mu_j)_{s_j}^{+} \right]^2 + \\
& \sum_{j=1}^{M} \left\{ \rho_j \left[(-\mu_j)_{s_j}^{+} \right]^2 - \frac{1}{2}\overline{LS} (\mu_j - \hat{\mu}_j)^2 \right\}
\end{aligned}$$

This immediately suggests us that if the sequences of sampling instants $\{T_j^L[\ell]\}_{\ell=0}^{\infty}$ satisfy the inequality in equation 16 for all $\ell = 0, 1, 2, ..., \infty$, and $j \in \mathcal{L}$, then $\dot{L}(x, s; \lambda, w) \leq 0$ is guaranteed for all t.

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

$$z_i = 0, \quad \forall i \in \mathcal{S}, \quad (-\mu_j)_{s_j}^{+} = 0, \quad \mu_j = \hat{\mu}_j, \quad \forall j \in \mathcal{L} \tag{19}$$

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

This theorem asserts that we need to select the transmit times $\{T_j^L[\ell]\}$ so that the inequality in equation 16 always hold. It could be easily done if we use the violation of the inequality to trigger the sampling and transmission of the link states. Each link j computes 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 $t = T_j^L[\ell]$, this error is zero and the inequality is trivially satisfied. After that time, $\mu_j(t)$ continues to change until the inequality is violated. 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 users $i \in \mathcal{S}_j$. We see that, unlike the primal algorithm in [13], which has interacting user events and link events, we only have link events here.

In theorem 4.1, the parameter ρ_j can be different for each link $j \in \mathcal{L}$. One may wonder why we do not simply choose $\rho_j = 1$ in equation 16. For stability consideration, it is correct that we can simply choose $\rho_j = 1$. However, as we will see in the next section, choosing a smaller ρ_j makes the network more robust to data dropouts. For this reason, we keep it as a parameter here.

We should point out that, in equation 16, if the positive projection stays active, in other words, $s_j(t) = 0$ and $\mu_j(t) > 0$ for t over some time interval, then the link dynamical system in equations 11-12 reduces to a memoryless function. In those situations, if we still use the inequality in equation 16 to trigger the link event, it would require that $\hat{\mu}_j(t) = \mu_j(t)$ over the interval. This is highly undesirable since it requires link j to sample and transmit its state infinitely fast. There are two possible solutions to this issue. One solution is, once the projection stays active, then we no longer use the primal-dual algorithm. Instead, we switch to the primal algorithm [13]. In this paper, we take a slightly simpler approach. To be specific, we are only interested in how fast the trajectory enters some neighborhood of the optimal point. This neighborhood is chosen large enough so that the positive projections will not stay active before entering the neighborhood. In the remaining part of the paper, we will assume that the positive projection in equation 16 cannot be active unless at the minimizer of $L(x, s; \lambda, w)$. This assumption is reasonable if we are only interested in converging to some neighborhood of the optimal point. It enables us to present and analyze the primal-dual algorithm in a much clearer way.

In the following work, we will find it convenient to use a slightly more conservative event than equation 16.

Corollary 4.2: Consider the Lagrangian in equation 7 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 $\lambda \geq 0$. For each link $j \in \mathcal{L}$, consider the sequence $\{T_j^L[\ell]\}_{\ell=0}^{\infty}$, and its dynamics satisfy equations 11-12. For each user $i \in \mathcal{S}$, let the user rate, $x_i(t)$, satisfy equation 15 with sampled link states defined in equation 14.

For all $j \in \mathcal{L}$, let ρ_j be a constant such that $0 < \rho_j \leq 1$. Assume that for all $j \in \mathcal{L}$, and all $\ell = 0, \cdots, \infty$ that

$$|\mu_j(t) - \hat{\mu}_j(t)| \leq \delta_j |\hat{\mu}_j(t)| \tag{20}$$

for $t \in [T_j^L[\ell], T_j^L[\ell+1])$, where δ_j is defined by

$$\delta_j = \sqrt{\frac{\rho_j}{\frac{1}{2}\overline{LS} + \rho_j}} \tag{21}$$

Then the user rates $x(t)$ asymptotically converge to the unique minimizer of $L(x, s; \lambda, w)$. ■

Proof: By the definition of δ_j in equation 21, equation 20 is equivalent to

$$\frac{1}{2}\overline{LS}|\mu_j(t) - \hat{\mu}_j(t)|^2 + \rho_j|\mu_j(t) - \hat{\mu}_j(t)|^2 \leq \rho_j|\hat{\mu}_j(t)|^2 \tag{22}$$

Therefore, we have

$$\begin{aligned} \frac{1}{2}\overline{LS}|\mu_j(t) - \hat{\mu}_j(t)|^2 &\leq \rho_j|\hat{\mu}_j(t)|^2 - \rho_j|\mu_j(t) - \hat{\mu}_j(t)|^2 \\ &\leq \rho_j|\mu_j(t)|^2 \end{aligned}$$

for all $t \in [T_j^L[\ell], T_j^L[\ell+1])$, $j \in \mathcal{L}$ and $\ell = 0, \cdots, \infty$. Since we assume that the positive projection in equation 16 cannot be active unless at the minimizer of $L(x, s; \lambda, w)$, all assumptions of theorem 4.1 are satisfied. We can thus conclude that $x(t)$ asymptotically converge to the unique minimizer of $L(x, s; \lambda, w)$. ■

The inequalities in equations 16 or 20 can both be used as the basis for the event-triggered algorithm. Equation 20 is a slightly more conservative condition, and we will use it in our **event-triggered distributed primal-dual algorithm** in the following.

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 user $i \in \mathcal{S}$ executes the following algorithm. It is continuously transmitting data at rate $x_i(t)$ at time t.

Algorithm 4.1: **User i's Update Algorithm**

1) **Parameter Initialization:** Set the initial user rate $x_i^0 > 0$. Let $T = 0$.
2) **State Initialization:** Wait for all neighbors $j \in \mathcal{L}_i$ to send their link states $\mu_j(T)$ and set $\hat{\mu}_j = \mu_j(T)$.
3) **Update User Rate:** Integrate the user rate equation

$$\begin{aligned} x_i(t) &= \int_T^t \left(\nabla U_i(x_i(\tau)) - \sum_{j \in \mathcal{L}_i} \hat{\mu}_j \right)_{x_i(\tau)}^+ d\tau \\ x_i(T) &= x_i^0 \end{aligned}$$

   where $t \in [T, T^+)$ and T^+ is the time instant when the following condition is true
   a) if user i receives a new link state $\mu_j^+(T^+)$ from link $j \in \mathcal{L}_i$, set $\hat{\mu}_j = \mu_j^+(T^+)$.
4) **Increment Time:** Set $T = T^+$, $x_i^0 = x_i^+(T^+)$ and go to step 3.

A similar algorithm is executed by all links $j \in \mathcal{L}$. The main assumption here is that link j can continuously monitor the link state $\mu_j(t)$ at any time $t \in \Re$.

Algorithm 4.2: **Link j's Update Algorithm**

1) **Parameter Initialization:** Set $T = 0$, $w_j > 0$, $0 < \rho_j \leq 1$, initial $s_j^0 \geq 0$, and δ_j is defined by

$$\delta_j = \sqrt{\frac{\rho_j}{\frac{1}{2}\overline{LS} + \rho_j}} \tag{23}$$

2) **State Initialization** Measure the local link state

$$\mu_j(T) = \frac{1}{w_j}\left(\sum_{i \in \mathcal{S}_j} x_i(T) - c_j + s_j^0 \right) \tag{24}$$

Transmit $\mu_j(T)$ to all users $i \in \mathcal{S}_j$ and set $\hat{\mu}_j = \mu_j(T)$.

3) **Link Update:** Integrate the equation

$$s_j(t) = \int_T^t (-\mu_j(\tau))^+_{s_j(\tau)}\, d\tau \quad (25)$$

$$\mu_j(t) = \frac{1}{w_j}\left(\sum_{i\in\mathcal{S}_j} x_i(t) - c_j + s_j(t)\right) \quad (26)$$

$$s_j(T) = s_j^0 \quad (27)$$

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

a) If $|\mu_j(t) - \hat{\mu}_j(t)| \geq \delta_j|\hat{\mu}_j(t)|$, then set $\hat{\mu}_j = \mu_j^+(T^+)$ and broadcast the updated link state $\mu_j^+(T^+)$ to all users $i \in \mathcal{S}_j$.

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

By corollary 4.2, the data rates $x(t)$ generated by algorithms 4.1-4.2 converge asymptotically to the unique minimizer of $L(x, s; \lambda, w)$, which is an approximate solution to the NUM problem.

V. Event-triggering with data dropouts

The discussion in the previous section did not consider data dropouts. To be specific, whenever the new sampled link state $\hat{\mu}_j(t)$ is obtained by link j, it is transmitted to the users $i \in \mathcal{S}_j$ successfully. This, however, does not necessarily happen in large scale networks. In this section, we take data dropouts into consideration, and gives an upper bound on the largest number of successive data dropouts each link can have, while ensuring the asymptotic convergence of the event-triggered algorithm in section IV. Using our result, each link can identify this upper bound for its subsystem in a decentralized way. We assume that data dropouts only happen when the sampled states $\hat{\mu}_j(t)$ are transmitted across the network.

First, let us see what happens when there are data dropouts in the network. Suppose link j detects a local event and obtains a new sampled state $\hat{\mu}_j$. Link j will then transmit the new $\hat{\mu}_j$ to users $i \in \mathcal{S}_j$. If the transmission fails, then users $i \in \mathcal{S}_j$ will not receive the new sampled link state. However, link j thinks the new state has been successfully transmitted, so in equation 20, $\hat{\mu}_j(t)$ has been updated to the new $\hat{\mu}_j$. This means users and links have different copies of the latest sampled link state, which may destabilize the system. Our main idea is, the event in equation 20 is a relatively conservative event if ρ_j is small, so even if some data dropouts happen, the system may still be stable.

Further discussion needs some additional notations. We use $r_j[k]$ to denote the time instant when link j samples its link state μ_j for the kth time (but not necessarily successfully transmitted), and use $T_j^L[\ell]$ to denote the time instant when the sampled state of link j has been successfully transmitted for the ℓth time. It is obvious that $\{T_j^L[\ell]\}_{\ell=0}^{\infty}$ is a subsequence of $\{r_j[k]\}_{k=0}^{\infty}$. Using these notations, the user dynamics in equation 15 and user's knowledge of the sampled link state in equation 14 still apply. Define error as $e_j(t) = \mu_j(t) - \hat{\mu}_j(T_j^L[\ell])$ on $t \in [T_j^L[\ell], T_j^L[\ell+1])$.

Theorem 5.1: Consider the Lagrangian in equation 7 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 $\lambda \geq 0$. For each link $j \in \mathcal{L}$, consider the sequences $\{T_j^L[\ell]\}_{\ell=0}^{\infty}$, $\{r_j[k]\}_{k=0}^{\infty}$, and its dynamics satisfy equations 11-12. For each user $i \in \mathcal{S}$, let the user rate, $x_i(t)$, satisfy equation 15 with sampled link states defined in equation 14.

For all $j \in \mathcal{L}$, let ρ_j be a constant such that $0 < \rho_j \leq 1$. Assume that for all $j \in \mathcal{L}$, and all $k = 0, \cdots, \infty$ that

$$|\mu_j(t) - \mu_j(r_j[k])| \leq \delta_j|\mu_j(r_j[k])| \quad (28)$$

for $t \in [r_j[k], r_j[k+1])$, where δ_j is defined by

$$\delta_j = \sqrt{\frac{\rho_j}{\frac{1}{2}\overline{LS} + \rho_j}} \quad (29)$$

Further assume that link j's largest number of successive data dropouts, $d_j \in \mathbb{Z}$, satisfies

$$d_j \leq D_j(\rho_j) = \log_{(\frac{1}{1-\delta_j})}(1 + \sqrt{\frac{2}{\overline{LS}}}) - 1 \quad (30)$$

then the user rates $x(t)$ asymptotically converge to the unique minimizer of $L(x, s; \lambda, w)$. ■

Proof: Consider link j over the time interval $[T_j^L[\ell], T_j^L[\ell+1])$. For notational convenience, we assume $T_j^L[\ell] = r_j[0] < r_j[1] < \cdots < r_j[d_j] < r_j[d_j+1] = T_j^L[\ell+1]$.

Consider $e_j(t)$ for any $t \in [r_j[k], r_j[k+1])$, we have

$$\begin{aligned} &|e_j(t)| = |\mu_j(t) - \hat{\mu}_j(T_j^L[\ell])| \\ \leq &\sum_{p=0}^{k-1} |\mu_j(r_j[p+1]) - \mu_j(r_j[p])| + |\mu_j(t) - \mu_j(r_j[k])| \end{aligned}$$

Applying equation 28 on the previous equation yields

$$|e_j(t)| \leq \delta_j \sum_{p=0}^{k} |\mu_j(r_j[p])| \quad (31)$$

for all $t \in [r_j[k], r_j[k+1])$.

From equation 28, we can easily obtain

$$|\mu_j(r_j[k])| \leq \frac{1}{1-\delta_j}|\mu_j(t)| \quad (32)$$

for all $t \in [r_j[k], r_j[k+1])$. Applying equation 28 repeatedly on $[r_j[k-1], r_j[k])$, $[r_j[k-2], r_j[k-1])$, $\cdots$, $[r_j[p], r_j[p+1])$, we know

$$|\mu_j(r_j[p])| \leq \left(\frac{1}{1-\delta_j}\right)^{k+1-p} |\mu_j(t)| \quad (33)$$

for all $t \in [r_j[k], r_j[k+1])$.

Applying equation 33 on equation 31 yields

$$|e_j(t)| \leq \delta_j \sum_{p=0}^{k} \left(\frac{1}{1-\delta_j}\right)^{k+1-p} |\mu_j(t)| \quad (34)$$

$$= \left[\left(\frac{1}{1-\delta_j}\right)^{k+1} - 1\right] |\mu_j(t)| \quad (35)$$

for all $t \in [r_j[k], r_j[k+1])$. This means for all $t \in [T_j^L[\ell], T_j^L[\ell+1])$, we have

$$|e_j(t)| \leq \left[\left(\frac{1}{1-\delta_j}\right)^{d_j+1} - 1\right]|\mu_j(t)| \quad (36)$$

Since the above inequality holds for all $\ell = 0, 1, \cdots, \infty$, we know that for all $t \geq 0$ we have

$$\begin{aligned} |\mu_j(t) - \hat{\mu}_j(t)| &\leq \left[\left(\frac{1}{1-\delta_j}\right)^{d_j+1} - 1\right]|\mu_j(t)| \\ &\leq \sqrt{\frac{2}{\overline{LS}}}|\mu_j(t)| \quad (37) \end{aligned}$$

The last inequality holds by applying equation 30.

From the proof of theorem 4.1 and apply equation 37, we know that for all $t \geq 0$ we have

$$\begin{aligned} & -\dot{L}(x, s; \lambda, w) \\ \geq\ & \frac{1}{2}\sum_{i=1}^{N} z_i^2 - \frac{1}{2}\sum_{j=1}^{M}\overline{LS}(\mu_j - \hat{\mu}_j)^2 + \sum_{j=1}^{M}\mu_j^2 \quad (38) \\ \geq\ & \frac{1}{2}\sum_{i=1}^{N} z_i^2 \quad (39) \end{aligned}$$

The convergence then follows easily. ■

d_j is link j's largest number of successive data dropouts, and $D_j(\rho_j)$ represents the the maximum allowable number of successive data dropouts for link j. This theorem guarantees that algorithm 4.1-4.2 still converges if each link j's largest number of successive data dropouts d_j does not exceed $D_j(\rho_j)$. However, it says nothing if this condition is not satisfied. We can easily see that $D_j(\rho_j)$ can be determined by link j itself locally. However, there is a tradeoff between $D_j(\rho_j)$ and the broadcast periods. In general, small ρ_j results in short broadcast periods and large $D_j(\rho_j)$, while large ρ_j results in long broadcast periods but small $D_j(\rho_j)$. Two extreme cases are, as $\rho_j \to 0$, $D_j(\rho_j) \to +\infty$, as $\rho_j \to 1$, $D_j(\rho_j) \to 0$.

VI. Broadcast period analysis for the primal-dual algorithm

This section presents results on the link broadcast period of the event-triggered primal-dual algorithm. We lower bound it as a function of the system states in the network.

Recall that $\{T_j^L[\ell]\}_{l=0}^{\infty}$ is the sequence of broadcast time instants for link $j \in \mathcal{L}$. We define the ℓth broadcast period of link j as

$$B_j[\ell] = T_j^L[\ell+1] - T_j^L[\ell] \quad (40)$$

Assume that the hypotheses in corollary 4.2 hold. This means link j will obtain a new sample of $\mu_j(t)$ when the inequality in equation 20 is about to be violated. Further assume that the gradient of the utility function $U_i(x_i)$ satisfies $\nabla U_i(x_i) \leq \beta$ for all $i \in \mathcal{S}$.

Consider the time interval $[T_j^L[\ell], T_j^L[\ell+1])$ for link j. Between the latest broadcast time, $T_j^L[\ell]$, and the next broadcast time, $T_j^L[\ell+1]$, it is quite possible that other links $\{j' | j' \in \mathcal{L}, j' \neq j\}$ may trigger one or multiple broadcasts. Define $\mathcal{N}_j$=$\{j' | j \in \mathcal{L}, j' \in \mathcal{L}, \mathcal{S}_j \bigcap \mathcal{S}_{j'} \neq \emptyset\}$, so $\mathcal{N}_j$ is the set of links that have at least one common user with link j. Notice that $\mathcal{N}_j$ includes link j itself. Assume that on time interval $[T_j^L[\ell], T_j^L[\ell+1])$, links in the set $\mathcal{N}_j - \{j\}$ have triggered a total of m broadcasts, and $b[p]$ is the time of the pth broadcast. For notational convenience, we assume $T_j^L[\ell] = b[0] < b[1] < \cdots < b[m] < b[m+1] = T_j^L[\ell+1]$. It is obvious that on $t \in [b[p], b[p+1])$, $p = 0, 1, \cdots, m$, $\hat{\mu}_{j'}(t)$ is fixed for all $j' \in \mathcal{N}_j$. Also define error as $e_j(t) = \mu_j(t) - \hat{\mu}_j(T_j^L[\ell])$ on $t \in [T_j^L[\ell], T_j^L[\ell+1])$.

We can then study the behavior of the error $e_j(t)$ on $t \in [b[p], b[p+1])$, where $p = 0, 1, \cdots, m$.

Define $\Omega = \{t \in [b[p], b[p+1]) | |e_j(t)| = 0\}$ On $t \in [b[p], b[p+1]) - \Omega$, we have

$$\begin{aligned} & \frac{d|e_j(t)|}{dt} \leq \left|\frac{de_j(t)}{dt}\right| = \left|\frac{d\mu_j(t)}{dt}\right| \\ =\ & \left|\frac{1}{w_j}\frac{ds_j(t)}{dt} + \frac{1}{w_j}\sum_{i\in\mathcal{S}_j}\frac{dx_i(t)}{dt}\right| \\ =\ & \Bigg|-\frac{1}{w_j}\mu_j(t) \\ & +\frac{1}{w_j}\sum_{i\in\mathcal{S}_j}\left(\nabla U_i(x_i(t)) - \sum_{j'\in\mathcal{L}_i}\hat{\mu}_{j'}(t)\right)_{x_i(t)}^{+}\Bigg| \\ =\ & \Bigg|-\frac{1}{w_j}e_j(t) - \frac{1}{w_j}\hat{\mu}_j(t) + \\ & \frac{1}{w_j}\sum_{i\in\mathcal{S}_j}\left(\nabla U_i(x_i(t)) - \sum_{j'\in\mathcal{L}_i}\hat{\mu}_{j'}(t)\right)_{x_i(t)}^{+}\Bigg| \\ \leq\ & \frac{1}{w_j}|e_j(t)| + \frac{1}{w_j}|\hat{\mu}_j(t)| + \frac{1}{w_j}\sum_{i\in\mathcal{S}_j}\nabla U_i(x_i(t)) \\ & +\frac{1}{w_j}\sum_{i\in\mathcal{S}_j}\sum_{j'\in\mathcal{L}_i}|\hat{\mu}_{j'}(t)| \\ \leq\ & \frac{1}{w_j}|e_j(t)| + \frac{1}{w_j}\beta\overline{S} \\ & +\frac{1}{w_j}|\hat{\mu}_j(T_j^L[\ell])| + \frac{1}{w_j}\sum_{i\in\mathcal{S}_j}\sum_{j'\in\mathcal{L}_i}|\hat{\mu}_{j'}(t)| \quad (41) \end{aligned}$$

where we use the right-hand sided derivative when $t = b[p]$. From the definition of $\mathcal{N}_j$, we know $\sum_{i\in\mathcal{S}_j}\sum_{j'\in\mathcal{L}_i}|\hat{\mu}_{j'}(t)|$ is constant on $t \in [b[p], b[p+1])$. We can then solve the differential inequality in equation 41, which gives us

$$\begin{aligned} |e_j(t)| \leq\ & e^{\frac{1}{w_j}(t-b[p])}|e_j(b[p])| + \Bigg(\beta\overline{S} + |\hat{\mu}_j(T_j^L[\ell])| \\ & + \sum_{i\in\mathcal{S}_j}\sum_{j'\in\mathcal{L}_i}|\hat{\mu}_{j'}(b[p])|\Bigg)\left[e^{\frac{1}{w_j}(t-b[p])} - 1\right] \quad (42) \end{aligned}$$

for all $t \in [b[p], b[p+1])$ since $|e_j(t)| = 0$ for all $t \in \Omega$.

Since link j's next broadcast will be triggered when the inequality in equation 20 is about to be violated. This will

happen at time $T_j^L[\ell+1]$ when

$$|e_j(T_j^L[\ell+1])| \geq \delta_j |\hat{\mu}_j(T_j^L[\ell])| \tag{43}$$

We can use the bound on $|e_j(t)|$ in equation 42 to solve for $T_j^L[\ell+1]$ in equation 43. It gives us

$$T_j^L[\ell+1] - b[m] \geq w_j \ln\Bigg(1 + \frac{\delta_j|\hat{\mu}_j(T_j^L[\ell])| - |e_j(b[m])|}{|e_j(b[m])| + \beta\overline{S} + |\hat{\mu}_j(T_j^L[\ell])| + Z_{\mathcal{N}_j}(b[m])}\Bigg)$$

where

$$Z_{\mathcal{N}_j}(b[m]) = \sum_{i\in\mathcal{S}_j}\sum_{j'\in\mathcal{L}_i} |\hat{\mu}_{j'}(b[m])| \tag{44}$$

The broadcast period of link j is then

$$\begin{aligned} B_j[\ell] &= T_j^L[\ell+1] - T_j^L[\ell] \\ &= T_j^L[\ell+1] - b[m] + \sum_{p=1}^{m}(b[p]-b[p-1]) \\ &\geq \sum_{p=1}^{m}(b[p]-b[p-1]) + w_j \ln\Bigg(1 + \frac{\delta_j|\hat{\mu}_j(T_j^L[\ell])| - |e_j(b[m])|}{|e_j(b[m])| + \beta\overline{S} + |\hat{\mu}_j(T_j^L[\ell])| + Z_{\mathcal{N}_j}(b[m])}\Bigg) \end{aligned}$$

The above equation provides a state-dependent lower bound for link j's broadcast period, $B_j[\ell]$. We can then conclude that the broadcast period of each link is bounded away from zero.

VII. Simulation

This section presents simulation results. It shows that our event-triggered primal-dual algorithm reduces the number of message exchanges by up to two order magnitude when compared to dual decomposition. Moreover, our algorithm is scale free with respect to network size. The robustness to data dropouts of our algorithm is also demonstrated. The remainder of this section is organized as follows: Subsection VII-A discusses the simulation setup. Simulation results on broadcast periods of our algorithm are shown in subsection VII-B. Subsection VII-C presents simulation results on our algorithm when data dropouts are taken into consideration. The scalability results are presented in subsection VII-D.

A. Simulation Setup

Denote $s \in \mathcal{U}[a,b]$ if s is a random variable uniformly distributed on $[a,b]$. Given M, N, $\overline{L}$ and $\overline{S}$, we randomly generate a network with M links and N users, where $|\mathcal{S}_j| \in \mathcal{U}[1,\overline{S}]$, $j \in \mathcal{L}$, $|\mathcal{L}_i| \in \mathcal{U}[1,\overline{L}]$, $i \in \mathcal{S}$. We make sure at least one link (user) has $\overline{S}$ users ($\overline{L}$ links). User i is assigned $U_i(x_i) = \alpha_i \log x_i$, where $\alpha_i \in \mathcal{U}[0.8, 1.2]$. Link j is assigned $c_j \in \mathcal{U}[0.8, 1.2]$. Both algorithms are simulated. The optimal rate x^* and corresponding U^* are calculated using a global optimization technique.

Define error as (for all algorithms)

$$e(k) = \left|\frac{U(x(k)) - U^*}{U^*}\right| \tag{45}$$

where $x(k)$ is the rate 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 from the links to the users occur at each iteration synchronously. So K is a measure of the total number of message exchanges. In our event-triggered algorithms, events occur in a totally asynchronous way. We add the total number of triggered events, and divide this number by the link number M. This works as an equivalent iteration number K for our event-triggered algorithms, and is a measure of the total number of message exchanges.

The default settings for simulation are as follows: $M = 60$, $N = 150$, $\overline{L} = 8$, $\overline{S} = 15$, $e_d = 3\%$. Initial condition is $x_i(0) \in \mathcal{U}[0.01, 0.05]$, $\forall i \in \mathcal{S}$. In dual decomposition, initial $p_j = 0$ for $j \in \mathcal{L}$, and the step size γ is calculated using equation 5. In our algorithm, $\rho_j = 0.9$, $\lambda_j = 0$, $w_j = 0.01$ for $j \in \mathcal{L}$.

B. Broadcast periods of the event-triggered algorithm

In this subsection we present simulation results on the broadcast periods of our algorithm.

This simulation uses the default settings in subsection VII-A and ran for 3.60s. The error in both algorithms entered the e_d neighborhood in $3.60s$. For reference, with the same settings, the average broadcast period for dual decomposition is 0.0026. In our algorithm, there are a total of 1971 link events. The links have an average broadcast period of 0.1096, which is 42 times longer than in dual decomposition.

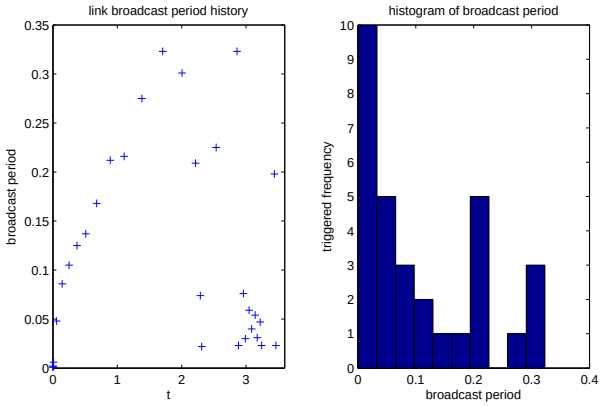

Fig. 1. Broadcast results for event-triggered algorithm

To see how the broadcast periods vary for a particular link in our algorithm, we pick one link in the network, and its broadcast results are shown in figure 1. The left plot is time history of broadcast periods generated by the link's local event. The right plot is the histogram of those link broadcast periods. The link's broadcast periods range between 0.0020 and 0.3230. The link was triggered 31 times, with an average broadcast period of 0.1161.

As we can see, our algorithm enjoys much longer broadcast periods than dual decomposition.

C. Data dropout simulation

In this subsection we present simulation results on our algorithm when data dropouts are taken into consideration.

Figure 2 plots the maximum allowable number of successive data dropouts $D_j(\rho_j)$ as a function ρ_j (in logarithm scale). As we can see, $D_j(\rho_j)$ is monotone decreasing in ρ_j. If we want the system to be robust to possible large data dropouts, then we need to choose a small ρ_j. For references, when $\rho = 0.208, 0.094, 0.024$, $D_j(\rho_j) = 1.0045, 2.0089, 5.0113$ respectively.

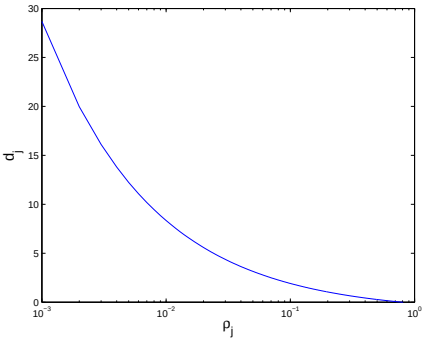

Fig. 2. Max allowable number of successive data dropouts $D_j(\rho_j)$ as a function of ρ_j

Table I summarizes simulation results for three different choices of ρ_j. For each given ρ_j, we use the same ρ_j, d_j for every link in the network. In the simulation, we assume that the number of successive data dropouts is always equal to the largest number of successive data dropouts, d_j. When $\rho_j = 0.094$ and $\rho_j = 0.024$, d_j is chosen so that the stability condition in equation 30 is satisfied. When $\rho_j = 0.9$, we intentionally choose $d_j = 10$ so that it violates the condition. The system is asymptotically stable in all three scenarios. This means that the bound on data dropout in equation 30 is indeed a sufficient condition for stability, but is a very conservative bound. Remember in subsection VII-B, when $\rho_j = 0.9$ and no data dropouts, the average broadcast period is 0.1096. This is close to the average successful broadcasts period in table I in all three cases. However, with no surprise, when data dropouts occur, the average triggered broadcasts period is much shorter than 0.1096, which is clearly shown in table I.

ρ_j	0.094	0.024	0.9
$D_j(\rho_j)$	2.0089	5.0113	0
d_j	2	5	10
Number of triggered broadcasts	7629	17342	18932
Number of successful broadcasts	2522	2867	1693
Average triggered broadcasts period	0.0283	0.0125	0.0114
Average successful broadcasts period	0.0856	0.0753	0.1276

TABLE I

SIMULATION RESULTS FOR DIFFERENT ρ_j AND d_j

D. Scalability with respect to $\overline{S}$ and $\overline{L}$

In this simulation we present simulation results on scalability of our algorithm.

In the first simulation, we fix M, N, $\overline{L}$ and vary $\overline{S}$ from 7 to 26. For each $\overline{S}$, both algorithms were run 250 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 $\overline{S}$. m_k works as our criteria for comparing the scalability of both algorithms. Figure 3 plots the iteration number K (in logarithm scale) as a function of $\overline{S}$ for both algorithms. The asterisks above represent m_K for dual decomposition, and the crosses below denote our primal-dual algorithm. The dotted vertical line around each asterisk and cross corresponds to the interval $[m_K - \sigma_K, m_K + \sigma_K]$ for each different $\overline{S}$ denoted by the x-axis. For our algorithm, when $\overline{S}$ increases from 7 to 26, m_K increases from 24.9 to 39.6, and σ_K increases from 0.6 and 3.4. As for dual decomposition, m_K increases from 0.3851×10^3 to 5.0899×10^3. σ_K at the same time increases from 0.4958×10^2 to 6.7957×10^2.

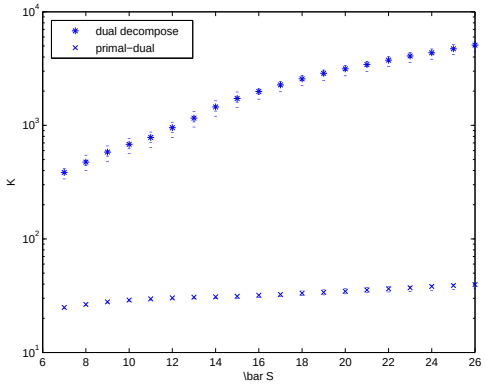

Fig. 3. Iteration number K as a function of $\overline{S}$ for both algorithms.

The second simulation is similar to the first one except that we vary $\overline{L}$ from 4 to 18 instead of $\overline{S}$. Figure 4 plots K (in logarithm scale) as a function of $\overline{L}$ for both algorithms. For our algorithm, when $\overline{L}$ increases from 4 to 18, m_K increases from 26.9 to 47.0, and σ_K varies between 1.1 and 3.7. As for dual decomposition, m_K increases from 0.9772×10^3 to 3.5174×10^3, and σ_K at the same time increases from 0.9668×10^2 to 6.1247×10^2.

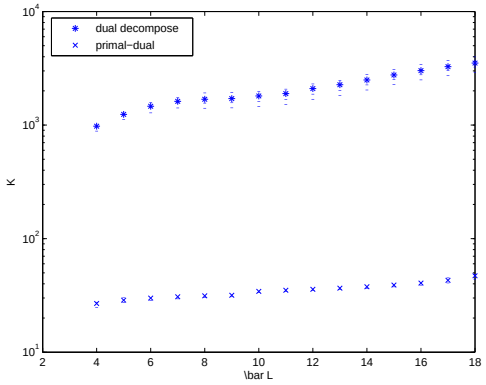

Fig. 4. Iteration number K as a function of $\overline{L}$ for both algorithms.

Our event-triggered algorithm is up to two order magnitude faster than the dual decomposition. We can also see that, unlike dual decomposition, which scales superlinearly with respect to $\overline{S}$ and $\overline{L}$, our event-triggered primal-dual algorithm on the other hand is scale-free. As shown above, the primal-

dual algorithm in this paper has comparable performance to the primal algorithm in [13].

VIII. CONCLUSION

This paper presents an event-triggered primal-dual algorithm for the NUM problem and proves its convergence. Results on the maximum allowable successive data dropouts and lower bound on broadcast periods are also given. Simulation results suggest that the algorithm is scale-free with respect to two measures of network size, and reduce the number of message exchanges by up to two orders of magnitude compared to dual decomposition.

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] 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.

[3] 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.

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

[5] 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.

[6] L. Xiao, M. Johansson, and S. Boyd, "Simultaneous routing and resource allocation via dual decomposition," *Communications, IEEE Transactions on*, vol. 52, no. 7, pp. 1136–1144, 2004.

[7] M. Chiang and J. Bell, "Balancing supply and demand of bandwidth in wireless cellular networks: utility maximization over powers and rates," *Proc. IEEE INFOCOM*, vol. 4, pp. 2800–2811, 2004.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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