Distributed Network Utility Maximization using Event-triggered Barrier Methods

Pu Wan and Michael D. Lemmon

Abstract— **Many problems associated with networked systems can be formulated as network utility maximization (NUM) problems. 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 dual decomposition scales poorly with respect to these measures. This paper investigates the use of an event-triggered communication scheme in distributed NUM algorithms. Under event triggering, each agent broadcasts to its neighbors when a local "error" signal exceeds a state dependent threshold. In particular, this paper proposes an event-triggered distributed NUM algorithm based on barrier methods. The paper establishes state-dependent event-triggering thresholds under which the proposed algorithm converges to the optimal solution of the NUM problem. Simulation results suggest that the proposed algorithm reduces 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. Many problems in networked systems, like distributed control of sensor-actuator networks [1], resource allocation in wireless communication networks [2] [3] and congestion control in wired communication networks [4] [5], 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 have been proposed to solve the NUM problem [5] [6] [7] after Kelly's seminar work [4]. Among all existing algorithms, the dual decomposition approach by Low et al. [5] 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 becomes extremely small. Step size 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 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 $\overline{S}$.

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).

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 systems, the energy required for communication can be significantly greater than the energy required to perform computation [8].

This paper presents one way of reducing the message passing complexity of distributed NUM algorithms. It has recently been demonstrated [9] that event-triggering in state feedback control systems can greatly lengthen the average sampling period of such systems. These results suggest that the use of event-triggering in a suitable NUM algorithm may significantly reduce the message passing complexity experienced by such algorithms. This paper presents a NUM algorithm based on barrier methods that uses event-triggered message passing. We prove that the proposed algorithm converges to the global optimal solution of the NUM problem. Simulations suggest that the resulting algorithm has a message passing complexity that is significantly lower than dual decomposition algorithms.

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 optimization algorithm is based on a barrier method solution to the NUM problem which is described in section III. Section IV presents our event-triggered distributed algorithm based on barrier methods, and proves its convergence. Simulation results are shown in section V, and section VI concludes the paper.

II. Dual Decomposition NUM Algorithm

NUM problem [4] 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 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

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

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 [4][5] user i gets by transmitting at rate x_i.

NUM problems are often solved using dual decomposition [5]. The algorithm examines the dual of the NUM problem, which is

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

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 [4]) 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. [5] 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

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

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

for $k = 0, \cdots, \infty$. It is showed 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. 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. Barrier Method NUM Algorithm

The event-triggered algorithm presented in this paper is based on a barrier method for the NUM problem. Barrier type algorithms have only recently been proposed for use on the NUM problem. This recent work uses a special type of barrier method known as the interior-point method [10]. Primal-dual interior point methods, however, do not distribute across the network. We therefore have to develop a barrier method that easily distributes across the network.

In barrier methods, a constrained problem is converted into a sequence of unconstrained problems, which involve an added high cost for approaching the boundary of the feasible region via its interior. The added cost is parameterized by the barrier parameter. As the barrier parameter decreases to zero, the added cost becomes increasingly inconsequential. This progressively allows the iterates to approach the optimal point on the boundary. As the barrier parameter goes to zero, the optimal point on the boundary is reached.

Traditional barrier algorithms [11] have only one barrier parameter. Our algorithms consider a more general case in which a barrier parameter is associated with each constraint. Let $\mathcal{F}^s = \{x : x > 0, Ax < c\}$ denote the strict feasible region. The Lagrangian associated with NUM problem is

$$\begin{aligned} L(x;\tau,\lambda) &= -\sum_{i\in\mathcal{S}} U_i(x_i) - \sum_{i\in\mathcal{S}}\lambda_i \log x_i \\ &\quad -\sum_{j\in\mathcal{L}}\tau_j \log(c_j - a_j^T x) \end{aligned} \tag{6}$$

where $\tau = [\tau_1, \cdots, \tau_M]$ is a vector of barrier parameters associated with the links in $\mathcal{L}$ and $\lambda = [\lambda_1, \cdots, \lambda_N]$ is a vector of barrier parameters associated with the users in $\mathcal{S}$. The vector $a_j^T = [A_{j1}, \cdots, A_{jN}]$ is the jth row of the routing matrix A.

Let $\{\overline{\tau}_j[k]\}_{k=0}^{\infty}$ and $\{\overline{\lambda}_i[k]\}_{k=0}^{\infty}$ be sequences of link ($j \in \mathcal{L}$) and user ($i \in \mathcal{S}$) barriers, respectively, that are monotone decreasing to zero. The barrier method solves the NUM problem by approximately minimizing $L(x;\overline{\tau}[k],\overline{\lambda}[k])$ for the barrier sequences defined above. Let $x^*[k]$ denote the approximate minimizer for $L(x;\overline{\tau}[k],\overline{\lambda}[k])$. By the barrier method in [11], the sequence of approximate minimizers $\{x^*[k]\}_{k=0}^{\infty}$ converges to the optimal point of the NUM problem. The algorithm is formally stated below.

1) **Initialization:** Select initial rate $x^0 \in \mathcal{F}^s$. Set $K = 0$, $\tau_j = \overline{\tau}_j[K]$, $\lambda_i = \overline{\lambda}_i[K]$, $i \in \mathcal{S}$, $j \in \mathcal{L}$, and $\epsilon = \overline{\epsilon}[K]$.
2) **Main Recursive Loop:**
   Do until: $\left\|\nabla_x L(x^0,\tau,\lambda)\right\| \leq \epsilon_d$
   a) **Approximately Minimize** $L(x;\tau,\lambda)$**:**
      Do until: $\left\|\nabla_x L(x^0;\tau,\lambda)\right\| \leq \epsilon$
      $$\begin{aligned} x &= x^0 - \gamma\nabla_x L(x^0;\tau,\lambda) \\ x^0 &= x \end{aligned} \tag{7}$$
   b) **Update Parameters:**
      Set $\tau_j = \overline{\tau}_j[K+1]$, $\lambda_i = \overline{\lambda}_i[K+1]$, $i \in \mathcal{S}$, $j \in \mathcal{L}$, and $\epsilon = \overline{\epsilon}[K+1]$. Set $K = K+1$.
3) Set $x^* = x^0$.

In the algorithm above, $\{\overline{\epsilon}[k]\}_{k=0}^{\infty}$ is a sequence of tolerance levels that are monotone decreasing to zero. ϵ_d is a terminating tolerance level. γ is a sufficiently small step size. Note that the inner recursion shown in step 2a is approximately minimizing $L(x;\tau,\lambda)$ for a fixed set of barrier vectors using a simple gradient following method.

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 barrier 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 is equal to the number of updates required for the algorithm's convergence. That number is determined by the step-size, which may be small, so that the number of messages passed between links and users will be large. The following section presents such an event-triggered implementation of the barrier method NUM algorithm.

IV. Event-triggered NUM Barrier Algorithm

Implementing the NUM barrier algorithm in a distributed manner requires communication between users and links. 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 problem's solution.

We begin by considering the minimization of $L(x;\overline{\tau}[k],\overline{\lambda}[k])$ for a *fixed* set of barrier vectors (i.e. fixed k). Subsection IV-A determines an event threshold condition ensuring the convergence of the local update (equation 7) to this minimizer. Subsection IV-B then considers the case when the barrier vectors are *switched* as we change k. In particular, we present a distributed update strategy for the parameters that ensures the convergence of the algorithm to the NUM problem's solution.

A. Fixed Barrier Parameter case

This subsection considers the problem of minimizing $L(x;\tau,\lambda)$ for fixed τ and λ. We can search for the minimizer using a gradient following algorithm

$$x_i(t) = \int_0^t \left(\frac{\partial U_i(x_i(s))}{\partial x_i} + \frac{\lambda_i}{x_i(s)} - \sum_{j\in\mathcal{L}_i} \mu_j(s) \right) ds \quad (8)$$

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

$$\mu_j(t) = \frac{\tau_j}{c_j - a_j^T x(t)} \quad (9)$$

Equation 8 is the continuous-time version of the update in equation 7. Note that in equation 8, user i 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. We can think of μ_j as the jth link's local *state*. From equation 9, link 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 user rate can be done in a distributed manner.

In the above equation, the link state information is available to the user in a continuous manner. We now consider an *event-triggered* version of equation 8. We assume that the user 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 link. The time $T_j^L[\ell]$ denotes the instant when the jth link samples its link state μ_j for the ℓth time and transmits that state to users $i \in \mathcal{S}_j$. We can therefore see that at any time $t \in \Re$, the sampled link state is a piecewise constant function of time in which

$$\hat{\mu}_j(t) = \mu_j(T_j^L[\ell]) \quad (10)$$

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 8 takes the form

$$x_i(t) = \int_0^t \left(\frac{\partial U_i(x_i(s))}{\partial x_i} + \frac{\lambda_i}{x_i(s)} - \sum_{j\in\mathcal{L}_i} \hat{\mu}_j(s) \right) ds \quad (11)$$

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

Define $z_i(t)$ as

$$z_i(t) = \dot{x}_i(t) = \nabla U_i(x_i(t)) + \frac{\lambda_i}{x_i(t)} - \sum_{j\in\mathcal{L}_i} \hat{\mu}_j(t) \quad (12)$$

for all $i \in \mathcal{S}$. We will refer to z_i as the ith *user state*. We associate a sequence $\{T_i^S[\ell]\}_{\ell=0}^\infty$ to each user $i \in \mathcal{S}$. The time $T_i^S[\ell]$ is the ℓth time when user i transmits its user state to all links $j \in \mathcal{L}_i$. At any time $t \in \Re$, the sampled user state is a piecewise constant function of time satisfying

$$\hat{z}_i(t) = z_i(T_i^S[\ell]) \quad (13)$$

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

We are now in a position to state the main theorem of this subsection. The proofs of all the theorems and lemmas in the paper will be found in the appendix.

Theorem 4.1: Consider the Lagrangian in equation 6 where U_i are twice differentiable, strictly increasing, and strictly concave and A is full rank. Assume fixed $\lambda > 0$, $\tau > 0$ and initial user rates $x_i(0) \in \mathcal{F}^s$. Consider the sequences $\{T_i^S[\ell]\}_{\ell=0}^\infty$ and $\{T_j^L[\ell]\}_{\ell=0}^\infty$ for each $i \in \mathcal{S}$, and each $j \in \mathcal{L}$, respectively. For each $i \in \mathcal{S}$, let the user rate, $x_i(t)$, satisfy equation 11 with sampled link states given by equation 10. For each $i \in \mathcal{S}$ let the user state $z_i(t)$ satisfy equation 12 and assume link j's measurement of the user state satisfies equation 13.

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

$$z_i^2(t) - \rho \hat{z}_i^2(t) \geq 0 \quad (14)$$

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

$$\rho \sum_{i\in\mathcal{S}_j} \frac{1}{\overline{L}} \hat{z}_i^2(t) - \overline{LS} \left(\mu_j(t) - \hat{\mu}_j(t)\right)^2 \geq 0 \quad (15)$$

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 $L(x;\tau,\lambda)$.

Theorem 4.1 provides the basis for constructing an event-triggered message-passing protocol. This theorem asserts that we can use the violation of the inequalities in equations 14 and 15 to trigger the sampling and transmission of link/user states across the network. At time $t = T_i^S[\ell]$, the inequality in equation 14 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 link. Similarly, link 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 link 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 user.

The threhold conditions shown in equations 14-15 therefore provide the basis for an event-triggered scheme to solve the local minimization problem in step 2a of the NUM barrier algorithm presented earlier.

B. The switching barrier parameter case

This subsection presents a distributed update strategy for the barrier parameters and proves the convergence of the resulting event-triggered distributed algorithm.

For a function $f(t)$ 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. We assume there exists monotone decreasing sequences of $\{\overline{\lambda}_i[k]\}_{k=0}^{\infty}$, and $\{\overline{\epsilon}_i[k]\}_{k=0}^{\infty}$ that asymptotically approaches zero.

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

1) **Parameter Initialization:** Set initial rate $x^0 \in \mathcal{F}^s$. Let $K = 0$, $T = 0$, $\lambda_i = \overline{\lambda}_i[K]$, and $\epsilon_i = \overline{\epsilon}_i[K]$.
2) **State Initialization:** Wait for all neighbors $j \in \mathcal{L}_i$ to send their link states $\mu_j(T)$ and set $\hat{\mu}_j = \mu_j(T)$. Initialize the user state, set $\hat{z}_i = z_i(T)$ and transmit $z_i(T)$ to all links in $j \in \mathcal{L}_i$.
3) **Update User Rate:** Integrate the user rate equation

$$x_i(t) = \int_T^t z_i(s)ds \quad (16)$$

$$z_i(t) = \nabla U_i(x_i(t)) + \frac{\lambda_i}{x_i(t)} - \sum_{j\in\mathcal{L}_i} \hat{\mu}_j \quad (17)$$

$$x_i(T) = x_i^0 \quad (18)$$

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 links $j \in \mathcal{L}_i$, and set $\hat{z}_i = z_i^+(T^+)$.
   b) Or if user i receives a new link state $\mu_j^+(T^+)$ from link $j \in \mathcal{L}_i$, set $\hat{\mu}_j = \mu_j^+(T^+)$.
   c) Or if $|z_i(t)| \leq \epsilon_i$, then set $\lambda_i = \overline{\lambda}_i[K+1]$ and $\epsilon_i = \overline{\epsilon}_i[K+1]$. Set $K = K+1$ and notify link $j \in \mathcal{L}_i$ that user i performed a barrier update.
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 $j \in \mathcal{L}$. Link j can continuously monitor the link state $\mu_j(t)$ at any time $t \in \Re$. We assume there exists a monotone decreasing sequence $\{\overline{\tau}_j[k]\}_{k=0}^{\infty}$ that asymptotically approaches zero.

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

1) **Parameter Initialization:** Set $K = 0$, $T = 0$, $\tau_j = \overline{\tau}_j[K]$ and set $I_i = 0$ for each $i \in \mathcal{S}_j$.
2) **State Initialization** Measure the local link state. Transmit $\mu_j(T)$ to all users $i \in \mathcal{S}_j$ and set $\hat{\mu}_j = \mu_j(T)$. Wait for users 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
$$\rho \sum_{i\in\mathcal{S}_j} \frac{1}{\overline{L}}\hat{z}_i^2 \leq \overline{LS}\left(\mu_j(t) - \hat{\mu}_j\right)^2$$
then set $\hat{\mu}_j = \mu_j^+(T^+)$ and broadcast the updated link state $\mu_j^+(T^+)$ to all users $i \in \mathcal{S}_j$.
   b) Or if link j receives a new user state $z_i^+(T^+)$ for any $i \in \mathcal{S}_j$, then set $\hat{z}_i = z_i^+(T^+)$.
   c) Or if link j receives notification that user i performed a barrier update, set $I_i = 1$.
4) **Update Barrier Parameter:** If $I_i = 1$ for all $i \in \mathcal{S}_j$, then set $\tau_j = \overline{\tau}_j[K+1]$, reset $I_i = 0$ for all $i \in \mathcal{S}_j$. Set $K = K+1$.
5) **Increment Time:** Set $T = T^+$ and go to step 3.

In the preceding algorithms, the parameters λ_i, τ_j, and ϵ_i are switched according to the parameter sequences $\{\overline{\lambda}_i[k]\}$, $\{\overline{\tau}_j[k]\}$, and $\{\overline{\epsilon}_i[k]\}$, respectively. These switches occur at discrete time instants. Provided an infinite number of switches occur, we can guarantee that the sequence of parameters used by the algorithm also asymptotically approach zero. The following lemma establishes that this occurs.

Lemma 4.2: In algorithms 4.1- 4.2. For each $i \in \mathcal{S}$, let $\{T_i^\lambda[k]\}_{k=0}^{M_i^S}$ denote the sequences of all time instants when λ_i and ϵ_i switch values. For each $j \in \mathcal{L}$, let $\{T_j^\tau[k]\}_{k=0}^{M_j^L}$ denote the sequence of all time instants when τ_j switches values. Then M_i^S and M_j^L are infinite for all i, j.

The following lemma provides a lower bound on $\dot{L}(x;\tau,\lambda)$ for fixed barrier parameters.

Lemma 4.3: Under the assumptions of theorem 4.1, for all $t \geq 0$,

$$-\frac{5}{2}\sum_{i\in\mathcal{S}} z_i^2(t) \leq \frac{dL(x(t);\tau,\lambda)}{dt} \leq 0 \quad (19)$$

We can now show that algorithms 4.1-4.2 asymptotically converge to the solution of the NUM problem.

Theorem 4.4: Under the assumptions of U_i, A, and ρ in theorem 4.1, the data rates $x(t)$ generated by algorithms 4.1-4.2 converge asymptotically to the unique solution of the NUM problem.

V. Simulation

This section presents simulation results.

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 that 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 capacity $c_j \in \mathcal{U}[0.8, 1.2]$. Both algorithms are simulated. The optimal rate x^* and its corresponding utility U^* are calculated using a global optimization technique.

Define error as (for both algorithms)

$$e(k) = \left|\frac{U(x(k)) - U^*}{U^*}\right| \quad (20)$$

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 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 algorithm, events occur asynchronously. We add the total number of triggered events and the number of message passings associated with the barrier parameter updates, and divide this number by M. 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 are as follows: $\rho = 0.5$, $e_d = 1\%$, $M = 60$, $N = 150$, $\overline{L} = 8$, $\overline{S} = 15$. Initial condition $x_i(0) = 0.95 \min_{j \in \mathcal{L}} c_j/N$, $\forall i \in \mathcal{S}$, which is kept in $\mathcal{F}^s$. Choosing the initial conditions can be done in a distributed way through message passings, however, we will not discuss this issue here. 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, $\{\overline{\lambda}_i[k]\}_{k=0}^{\infty}$, $\{\overline{\epsilon}_i[k]\}_{k=0}^{\infty}$, $\{\overline{\tau}_j[k]\}_{k=0}^{\infty}$ are chosen as $\{0.1^k\}_{k=0}^{\infty}$, $\{5 \times 0.1^k\}_{k=0}^{\infty}$, $\{0.1^k\}_{k=0}^{\infty}$, respectively.

B. *Scalability with respect to $\overline{S}$*

In this simulation, we fix M, N, $\overline{L}$ and vary $\overline{S}$ from 7 to 26. For each $\overline{S}$, both algorithms were run 300 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 the two algorithms. Figure 1 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, 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 $\overline{S}$ denoted by the x-axis.

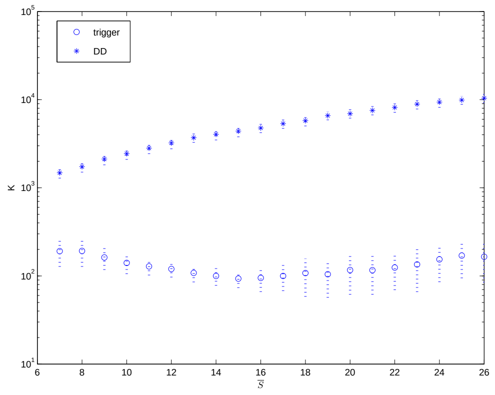

Fig. 1. Iteration number K as a function of $\overline{S}$ for both algorithms.

For our event-triggered algorithm, when $\overline{S}$ increases from 7 to 26, m_K does not show noticeable increase, and it varies between 93 and 192. The standard deviation σ_K varies between 20 and 80. For dual decomposition, m_K increases from 0.148×10^4 to 1.040×10^4. σ_K at the same time increases from 0.197×10^3 to 1.222×10^3. 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 $\overline{S}$, our algorithm on the other hand is virtually scale-free.

C. *Scalability with respect to $\overline{L}$*

This simulation is similar to subsection V-B except that we vary $\overline{L}$ from 4 to 18 instead of $\overline{S}$. Figure 2 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 does not show noticeable increase, and it varies between 95 and 224. σ_K varies between 24 and 88. We notice that, however, our event-triggered algorithm has worse performance when $\overline{L}$ is small. For dual decomposition, m_K increases from 1.774×10^3 to 8.631×10^3. σ_K at the same time increases from 0.183×10^3 to 1.26×10^3. Our event-triggered algorithm is up to two order magnitude faster than the dual decomposition, and is virtually scale-free with respect to $\overline{L}$.

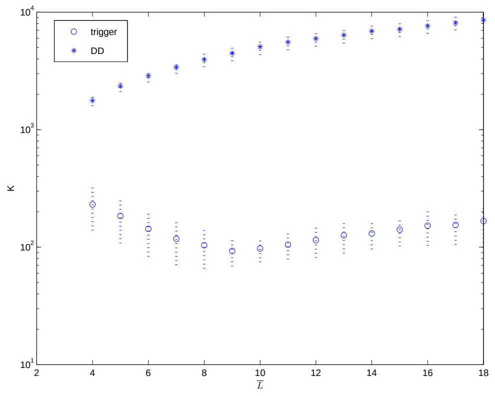

Fig. 2. Iteration number K as a function of $\overline{L}$ for both algorithms.

The reason the event-triggered algorithm has worse performance at small $\overline{L}$ is beacuse the parameters we choose work poorly for those cases. The choice of barrier parameters can sometimes greatly affect the performance of our algorithm. We will investigate how to optimally choose these parameters in future work.

VI. Conclusion

This paper presents an event-triggered distributed NUM algorithm based on the barrier methods. Simulation results suggest that the proposed algorithm is scale-free with respect to two measures of network size, and reduces the number of message exchanges by up to two orders of magnitude when compared to existing dual decomposition algorithms.

VII. Appendix

A. *Proof of Theorem 4.1*

Proof: For convenience, we do not explicitly include time dependence in the proof. For all $t \geq 0$ we have

$$-\dot{L}(x;\tau,\lambda) = \sum_{i=1}^{N} z_i[\nabla U_i(x_i) - \sum_{j=1}^{M} \mu_j A_{ji}]$$

$$\geq \sum_{i=1}^{N} \left\{ \frac{1}{2} z_i^2 - \frac{1}{2} [\sum_{j=1}^{M} (\mu_j - \hat{\mu}_j) A_{ji}]^2 \right\} \quad (21)$$

$$\geq \frac{1}{2}\sum_{i=1}^{N} z_i^2 - \frac{1}{2}\sum_{i=1}^{N} \left\{ |\mathcal{L}_i| \sum_{j=1}^{M} [(\mu_j - \hat{\mu}_j) A_{ji}]^2 \right\} \quad (22)$$

$$\geq \frac{1}{2}\sum_{i=1}^{N} z_i^2 - \frac{1}{2}\sum_{j=1}^{M} \overline{LS}(\mu_j - \hat{\mu}_j)^2 \quad (23)$$

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

$$\frac{1}{2}\rho\sum_{i=1}^{N}\hat{z}_i^2 = \frac{1}{2}\rho\sum_{i=1}^{N}\overline{L}\frac{1}{\overline{L}}\hat{z}_i^2 \quad (24)$$

$$= \frac{1}{2}\rho\sum_{j=1}^{M}\sum_{i=1}^{N}\frac{1}{\overline{L}}\hat{z}_i^2 A_{ji} + \frac{1}{2}\rho\sum_{i=1}^{N}(\overline{L} - |\mathcal{L}_i|)\frac{1}{\overline{L}}\hat{z}_i^2 \quad (25)$$

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

$$-\dot{L}(x;\tau,\lambda) \geq \frac{1}{2}\sum_{i=1}^{N}[z_i^2 - \rho\hat{z}_i^2] +$$

$$\frac{1}{2}\sum_{j=1}^{M}\left\{ \rho\sum_{i\in\mathcal{S}_j}\frac{1}{\overline{L}}\hat{z}_i^2 - \overline{LS}(\mu_j - \hat{\mu}_j)^2 \right\} \quad (26)$$

This means if the inequalities in equations 14 and 15 hold, then $\dot{L}(x;\tau,\lambda) \leq 0$ is guaranteed for all t. Using the properties of U_i, we know for any fixed τ and λ, $L(x;\tau,\lambda)$ has an unique minimizer $x^*(\tau,\lambda)$. Suppose the corresponding Lagrangian is $L(x^*;\tau,\lambda)$ and define $V(x) = L(x;\tau,\lambda) - L(x^*;\tau,\lambda)$. It is trivial to see $V(x)$ is a Lyapunov function for the system. Moreover, $\dot{V}(x) = 0$ means $\dot{L}(x;\tau,\lambda) = 0$, this can only happen at $x^*(\tau,\lambda)$. As a result, $x^*(\tau,\lambda)$ is asymptotically stable. Proof complete. ■

B. Proof of Lemma 4.2

Proof: We will first show that M_i^S is infinite for all $i \in \mathcal{S}$, then show M_j^L is infinite for all $j \in \mathcal{L}$.

Remember λ_i and ϵ_i are switched according to $\{\overline{\lambda}_i[k]\}_{k=0}^{\infty}$ and $\{\overline{\epsilon}_i[k]\}_{k=0}^{\infty}$. This means after finite or infinite switches, they will converge to their equilibria λ_i^*, ϵ_i^* respectively [12]. Assume λ_i and ϵ_i are at the equilibrium, then by the algorithm, for each link j, τ_j is also at its equilibrium. This means we now have fixed $\tau^*, \lambda^*, \epsilon^*$. Suppose at least one user r has a nonzero equilibrium ϵ_r^*. From theorem 4.1, $z_r(t)$ enters the $|z_r(t)| \leq \underline{\epsilon}_r$ neighborhood in finite time for any $\underline{\epsilon}_r > 0$. If we choose $\underline{\epsilon}_r$ to be any element in the sequence $\{\overline{\epsilon}_r[k]\}_{k=0}^{\infty}$ that is smaller than ϵ_r^*, then a user switch will occur for user r according to the algorithm, which contradicts the assumption that ϵ_r is already at its equilibrium. This means $\epsilon_i^* = 0$, $\forall i \in \mathcal{S}$. As a result, we know for each user i, M_i^S is infinite.

Next we show for each link j, M_j^L is also infinite. Define $T^{(0)} = \max_{i\in\mathcal{S}} T_i^{\lambda}[0]$. Then on $t \in [0, T^{(0)}]$, each user $i \in \mathcal{S}$ has completed at least one switch. By algorithm 4.2, each link $j \in \mathcal{L}$ also has completed at least one switch. Repeatedly using this argument, we can partition $[0,+\infty)$ into $\bigcup_{k=0}^{\infty}[T^{(k)}, T^{(k+1)})$. On each time interval, each link $j \in \mathcal{L}$ has completed at least one switch. Since M_i^S is infinite for each i, there are infinite such intervals. This means M_j^L is also infinite for each $j \in \mathcal{L}$. Proof complete. ■

C. Proof of Lemma 4.3 (Can be easily shown.)

D. Proof of Theorem 4.4

Proof: By lemma 4.2, there are infinite user switches for each user. Let $\{T[k]\}_{k=0}^{\infty}$ be the sorted sequence of all the elements in $\{T_i^{\lambda}[k]\}_{k=0}^{M_i^S}$ for all $i \in \mathcal{S}$. We can partition $[0,+\infty)$ into $\bigcup_{k=0}^{\infty}[T[k], T[k+1])$. Here $T[k]$ is the time instant when a user barrier switch occurs for any user. On $[T[k], T[k+1])$, we have fixed τ, λ, ϵ. By lemma 4.3,

$$|z_i(t)| \leq \tilde{\epsilon}_i(t), -\frac{5}{2}\sum_{i=1}^{N}\tilde{\epsilon}_i^2(t) \leq -\frac{5}{2}\sum_{i=1}^{N} z_i^2(t) \leq \dot{L}(x;\tau,\lambda) \leq 0$$

where $\tilde{\epsilon}_i(t) = \overline{\epsilon}_i[k-1]$, $t \in [T_i^{\lambda}[k], T_i^{\lambda}[k+1])$.

Define $f(t) = -\frac{5}{2}\sum_{i=1}^{N}\tilde{\epsilon}_i^2(t)$, Also $\forall i \in \mathcal{S}$, define $g_i(t) = \tilde{\epsilon}_i(t)$. Then $f(t)$ and $g_i(t)$ are nondecreasing functions of t that converges to 0. This means $|z_i(t)|$ and $\dot{L}(x;\tau,\lambda)$ converge to zero as $t \to \infty$. From equation 26 and the user and link events in algorithms 4.1-4.2, we get

$$\lim_{t\to\infty}\{-\frac{\partial L}{\partial x_i}\} = \lim_{t\to\infty} z_i(t) - \lim_{t\to\infty}\sum_{j\in\mathcal{L}_i}(\mu_j(t) - \hat{\mu}_j(t)) = 0$$

Since $\frac{\partial L}{\partial x_i}$ is a continuous function of x, we have $\lim_{t\to\infty} x_i(t) = x_i^*$,$\forall i \in \mathcal{S}$. This completes the proof. ■

References

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

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] X. Wang and M. Lemmon, "Event-triggered Broadcasting across Distributed Networked Control Systems," in *Proceedings of the American Control Conference*, 2008.

[10] A. Zymnis, N. Trichakis, S. Boyd, and D. ONeill, "An Interior-Point Method for Large Scale Network Utility Maximization," *Proceedings of the Allerton Conference*, 2007.

[11] J. Nocedal and S. Wright, *Numerical Optimization*. Springer, 1999.

[12] R. Walter, *Principles of mathematical analysis*. McGraw-Hill, 1976.