Distributed Network Utility Maximization using Event-triggered augmented Lagrangian methods

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. Motivated by recent results with event-triggered state feedback control, 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 the augmented Lagrangian 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 show that the proposed algorithm reduces the number of message exchanges by 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. Many problems in networked systems, like distributed control of sensor-actuator networks [1], resource allocation in wireless communication networks [2] [3] [4] [5], and congestion control in wired communication networks [6] [7], 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. In all of these problems, we find subsystems communicating with each other in order to collaboratively solve a network optimization problem.

A variety of distributed algorithms have been proposed to solve the NUM problem [6] [7] [8] [9]. Kelly [6] 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. [7] 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 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 [10]. As a result, there is significant interest in finding algorithms that reduce the message passing complexity of distributed optimization algorithms such as dual decomposition.

This paper presents one way of reducing the message passing complexity of distributed NUM algorithms. It has recently been demonstrated [11] [12] 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 the augmented Lagrangian methods that uses event-triggered message passing. We prove that the proposed algorithm converges to the global optimal solution of the NUM problem. Simulation experiments suggest that the resulting algorithm has a message passing complexity that is two orders of magnitude lower than dual decomposition algorithms, and is scale-free with respect to the above two measures of network size. This work is similar to our prior work in [13]. However, in [13], we used the barrier method instead of the augmented Lagrangian method as the basis for the event-triggered algorithm. In that case, the resulting algorithm suffers from issues like ill-conditioning and the need for an initial feasible point. The event-triggered algorithm presented in this paper does not have these issues.

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 an augmented Lagrangian method so-

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).

lution to the NUM problem which is described in section III. Section IV presents our event-triggered distributed algorithm based on the augmented Lagrangian 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 [6] 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) [6][7] user i gets by transmitting at rate x_i.

NUM problems are often solved using the dual decomposition algorithm [7]. 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 [6]) 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. [7] 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 the initial user rates $x[0]$ and link prices $p[0]$, 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 $k = 0, \cdots, \infty$.

The step size γ in equation 4 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. [7] showed that a suitable step size is

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

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. 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.

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

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

Here a penalty parameter w_j is associated with each link constraint, and $w = [w_1, \cdots, w_M]$ is the vector of penalty parameters. Suppose λ_j^* is the Lagrange multiplier associated with link j's 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 $\lambda = [\lambda_1, \cdots, \lambda_M]$. 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 [14] that

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

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

$$L(x; \lambda, w) = -\sum_{i\in\mathcal{S}} U_i(x_i) + \sum_{j\in\mathcal{L}} \psi_j(x; \lambda, w) \tag{8}$$

where

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

The augmented Lagrangian method solves the NUM problem by approximately 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 [14, 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 link ($j \in \mathcal{L}$) penalty parameters that are monotone decreasing to zero. $\{\lambda_j[k]\}_{k=0}^{\infty}$ are sequences of Lagrange multiplier estimates, where $\lambda_j[k+1] = \max\{0, \lambda_j[k] + \frac{1}{\overline{w}_j[k]}(a_j^T x^*[k] - c_j)\}$. The augmented Lagrangian method algorithm for the NUM problem is formally stated below.

1) **Initialization:** Select any initial user rate $x^0 > 0$, vector $\lambda \geq 0$ and set $K = 0$. Set $w_j = \overline{w}_j[K]$, $j \in \mathcal{L}$, and $\epsilon = \overline{\epsilon}[K]$.
2) **Main Recursive Loop:**
   Do until: $\left\|\nabla_x L(x^0; \lambda, w)\right\| \leq \epsilon_d$
   a) **Approximately minimize** $L(x; \lambda, w)$**:**
      Do until: $\left\|\nabla_x L(x^0; \lambda, w)\right\| \leq \epsilon$

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

   b) **Update Parameters:**
      Set $w_j = \overline{w}_j[K+1]$, $\lambda_j = \max\{0, \lambda_j + \frac{1}{w_j}(a_j^T x - c_j)\}$, $j \in \mathcal{L}$, 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 $L(x; \lambda, w)$ for fixed λ and w using a simple gradient following method. It is known [14] that by using the update of λ in step 2b, λ will converge to λ^*. We know that when w is decreased to zero, the algorithm converges. However, with an increasingly accurate estimate λ, it usually suffices if w is smaller than some threshold [14]. In this way, we may not need to decrease w to zero in practice, and thereby alleviating the difficulty with ill-conditioning when w is small.

The computations above can be easily distributed among the users and links. The primary computations that need to be distributed are the user rate update and terminating condition in step 2a, as well as the parameter update in step 2b. We will see how they are distributed in our event-triggered distributed implementation of the algorithm in section IV.

In dual decomposition and the augmented Lagrangian method 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 links and users 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 between links and users will be large.

Recent results [11] [12] show that it is possible to greatly reduce the message passing complexity of networked control systems by using *event-triggered* messages. The following section presents an event-triggered distributed implementation of the augmented Lagrangian method NUM algorithm.

IV. Event-triggered NUM augmented Lagrangian Algorithm

The NUM augmented Lagrangian algorithm solves a sequence of unconstrained optimization problems that are indexed with respect to the non-negative integers k. In particular, the algorithm minimizes the Lagrangian $L(x; \lambda[k], \overline{w}[k])$ where $\{\overline{w}[k]\}_{k=0}^{\infty}$ are sequences of link penalty parameters, that are monotone decreasing to zero. The vector $\lambda[k]$ is computed after the $(k-1)$th minimization problem.

Implementing the NUM augmented Lagrangian algorithm 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 NUM augmented Lagrangian algorithm to the problem's solution.

We begin by considering the minimization of $L(x; \lambda[k], \overline{w}[k])$ for a *fixed* set of parameters (i.e. fixed k). Subsection IV-A determines an event threshold condition ensuring the convergence of the local update (equation 9) to this minimizer. Subsection IV-B then considers the case when the penalty parameters are *switched* as we change k. In particular, we present a distributed update strategy for the penalty parameters that ensures the convergence of the algorithm to the NUM problem's solution.

A. Fixed penalty parameter case

This subsection considers the problem of finding a minimizer for the Lagrangian $L(x; \lambda, w)$ under the assumption that the vectors w and λ are constant. We can search for the minimizer using a gradient following algorithm

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

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

$$\mu_j(t) = \max\left\{0, \lambda_j + \frac{1}{w_j}(a_j^T x(t) - c_j)\right\} \quad (11)$$

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

$$(f(x))^+_x = \begin{cases} 0, & \text{if} \quad x = 0 \quad \text{and} \quad f(x) < 0 \\ f(x), & \text{otherwise} \end{cases} \quad (12)$$

The positive projection used in equation 10 guarantees the user rate $x_i(t)$ is always nonnegative along the trajectory.

Equation 10 is the continuous-time version of the update in equation 9. Note that in equation 10, 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 11, 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, this link state information is available to the user in a continuous manner. We now consider an *event-triggered* version of equation 10. Here 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 see that at any time $t \in \Re$, the sampled link state is a piecewise constant function of time in which

$$\hat{\mu}_j(t) = \mu_j(T_j^L[\ell]) \tag{13}$$

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 10 takes the form

$$x_i(t) \quad = \quad \int_0^t \left(\frac{\partial U_i(x_i(s))}{\partial x_i} - \sum_{j \in \mathcal{L}_i} \hat{\mu}_j(s) \right)_{x_i(s)}^{+} ds \tag{14}$$

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 link transmits its "state" to its users. Under event-triggering, it will be convenient to have a similar flow of information from the user to the link. We assume that link 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 link j have knowledge of the time derivative of user i's flow rate. In particular, let $z_i(t)$ denote the time derivative of this flow rate. $z_i(t)$ therefore satisfies

$$z_i(t) \quad = \quad \dot{x}_i(t) = \left(\nabla U_i(x_i(t)) - \sum_{j \in \mathcal{L}_i} \hat{\mu}_j(t) \right)_{x_i(t)}^{+} \tag{15}$$

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 $j \in \mathcal{L}_i$. We can therefore see that at any time $t \in \Re$, the sampled user state is a piecewise constant function of time satisfying

$$\hat{z}_i(t) = z_i(T_i^S[\ell]) \tag{16}$$

for all $\ell = 0, \cdots, \infty$ and any $t \in [T_i^S[\ell], T_i^S[\ell+1])$. In the proposed event-triggering scheme, links will use the sampled user state, $\hat{z}$, to help determine when they should transmit their states back to the user.

Next we will 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 8 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 $\lambda \geq 0$. 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 14 with sampled link states given by equation 13. For each $i \in \mathcal{S}$ let the user state $z_i(t)$ satisfy equation 15 and assume link j's measurement of the user state satisfies equation 16.

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

$$z_i^2(t) - \rho \hat{z}_i^2(t) \geq 0 \tag{17}$$

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

$$\rho \sum_{i \in \mathcal{S}_j} \frac{1}{L} \hat{z}_i^2(t) - \overline{LS} \left(\mu_j(t) - \hat{\mu}_j(t) \right)^2 \geq 0 \tag{18}$$

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 $L(x; \lambda, w)$. ■

Theorem 4.1 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 17 and 18 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 17 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 threshold conditions shown in equations 17-18 provide the basis for an event-triggered scheme to solve the local minimization problem in step 2a of the NUM augmented Lagrangian algorithm presented earlier.

B. The switching penalty parameter case

We now consider what happens when we systematically reduce these penalty parameters w to zero. In particular, we need to identify a distributed strategy for updating these penalty parameters so that the sequence of approximate minimizers asymptotically approach the global solution of the NUM problem. This subsection presents such an updating strategy and proves that the resulting event-triggered algorithm asymptotically converges to the desired solution.

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. The main assumption here is that user i is continuously transmitting data at rate $x_i(t)$ at time t. For each user i, we assume

there exists a monotone decreasing sequence of tolerance levels, $\{\overline{\epsilon}_i[k]\}_{k=0}^{\infty}$ that asymptotically approaches zero.

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

1) **Parameter Initialization:** Set the initial user rate $x_i^0 > 0$. Let $K = 0$, $T = 0$, 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 to
$$z_i(T) = \left(\nabla U_i(x_i^0) - \sum_{j \in \mathcal{L}_i} \hat{\mu}_j \right)^+_{x_i(T)} \quad (19)$$
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
$$x_i(t) = \int_T^t z_i(s)ds \quad (20)$$
$$z_i(t) = \left(\nabla U_i(x_i(t)) - \sum_{j \in \mathcal{L}_i} \hat{\mu}_j \right)^+_{x_i(t)} \quad (21)$$
$$x_i(T) = x_i^0 \quad (22)$$
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 $\epsilon_i = \overline{\epsilon}_i[K+1]$. Set $K = K + 1$ and notify link $j \in \mathcal{L}_i$ that user i performed a tolerance 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 $j \in \mathcal{L}$. The main assumption here is that link j can continuously monitor the link state $\mu_j(t)$ at any time $t \in \Re$. For each link j we assume there exists a monotone decreasing sequence of link penalty parameters, $\{\overline{w}_j[k]\}_{k=0}^{\infty}$ that asymptotically approaches zero.

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

1) **Parameter Initialization:** Set $K = 0$, $T = 0$, $w_j = \overline{w}_j[K]$, set $\lambda_j \geq 0$ and set the switching indicator $I_i = 0$ for each $i \in \mathcal{S}_j$.
2) **State Initialization** Measure the local link state
$$\mu_j(T) = \max \left\{ 0, \lambda_j + \frac{1}{w_j} \left(\sum_{i \in \mathcal{S}_j} x_i(T) - c_j \right) \right\} \quad (23)$$
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 tolerance update, set $I_i = 1$.
4) **Update Penalty Parameter:** If $I_i = 1$ for all $i \in \mathcal{S}_j$, then set $w_j = \overline{w}_j[K+1]$, $\lambda_j = \mu_j^+(T^+)$, 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 w_j and ϵ_i are switched according to the sequences $\{\overline{w}_j[k]\}$, and $\{\overline{\epsilon}_i[k]\}$, respectively. The convergence of the algorithms relies on decreasing w_j and ϵ_i to zero. However, the switches of λ_j in the link algorithm will in general help increase the convergence speed. These switches occur at discrete instants in time. Provided an infinite number of switches occur, we can guarantee that the sequence of parameters w_j and ϵ_i used by the algorithms also asymptotically approach zero. The following lemma establishes that this actually occurs.

Lemma 4.2: Consider algorithms 4.1 and 4.2. For each $i \in \mathcal{S}$, let $\{T_i^{\epsilon}[k]\}_{k=0}^{M_i^S}$ denote the sequences of all time instants when ϵ_i switch values. For each $j \in \mathcal{L}$, let $\{T_j^{w}[k]\}_{k=0}^{M_j^L}$ denote the sequence of all time instants when w_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 $\dot{L}(x; \lambda, w)$ for fixed penalty parameters. This bound will be later used in the proof of theorem 4.4.

Lemma 4.3: Under the assumptions of theorem 4.1, for all $t \geq 0$,
$$-\frac{5}{2} \sum_{i \in \mathcal{S}} z_i^2(t) \leq \frac{dL(x(t); \lambda, w)}{dt} \leq 0 \quad (24)$$

We can now show that algorithms 4.1-4.2 asymptotically converge to the solution of NUM. The main idea is to divide the entire time axis into an infinite number of mutually disjoint time intervals. On each interval, we have fixed penalty parameters. Since there are an infinite number of penalty parameter switches, we can show that the bound in lemma 4.3 asymptotically converges to zero, thereby showing that $\dot{L}$ converges to zero and thus establishing the convergence of the algorithm.

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. We compare the number of message exchanges of our event-triggered algorithm against the dual decomposition algorithm. 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 is scale free with respect to network size. The remainder of this section is organized as follows: Subsection V-A discusses the

simulation setup. The scalability results with respect to $\overline{S}$ and $\overline{L}$ are presented in subsection V-B and V-C, respectively.

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 $\overline{S}$, the network used for simulation is generated in the following way. 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 has $\overline{S}$ users, and at least one user uses $\overline{L}$ links. After the network is generated, we assign utility function $U_i(x_i) = \alpha_i \log x_i$ for each user i, where $\alpha_i \in \mathcal{U}[0.8, 1.2]$. Link j is assigned capacity $c_j \in \mathcal{U}[0.8, 1.2]$. Once the network is generated, 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)

$$e(k) = \left| \frac{U(x(k)) - U^*}{U^*} \right| \tag{25}$$

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, link events and user events occur in a totally asynchronous way. We add the total number of triggered events and the number of message passings associated with the penalty parameter updates, and divide this number by the link number 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 for simulation are as follows: For both algorithms, the initial condition $x_i(0) \in \mathcal{U}[0.01, 0.05]$, $\forall i \in \mathcal{S}$. In dual decomposition, initial price $p_j = 0$ for $j \in \mathcal{L}$, and the step size γ is calculated using equation 5. In our event-triggered algorithm, the initial Lagrange multiplier estimate is $\lambda_j = 0$ for $j \in \mathcal{L}$. The sequences $\{\overline{\epsilon}_i[k]\}_{k=0}^{\infty}$ and $\{\overline{w}_j[k]\}_{k=0}^{\infty}$ are chosen as $\{0.1^k\}_{k=0}^{\infty}$ and $\{0.01 \times 0.1^k\}_{k=0}^{\infty}$, respectively. Other parameters include $\rho = 0.5$, $e_d = 1\%$, $M = 60$, $N = 150$, $\overline{L} = 8$, $\overline{S} = 15$.

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 2000 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.

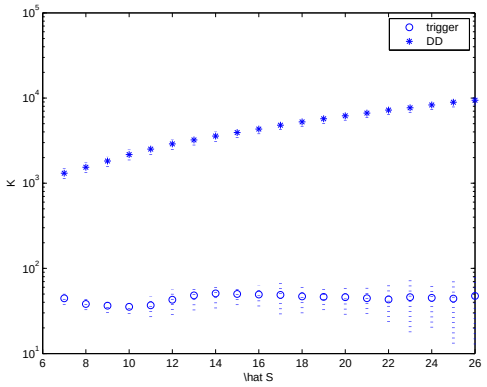

Fig. 1. Iteration number K as a function of $\overline{S}$ for both algorithms.

For our event-triggered algorithm, when $\overline{S}$ increases from 7 to 26, m_K does not show noticeable increase, and it varies between 36 and 50. σ_K varies between 6 and 34. For dual decomposition, m_K increases from 1.3119×10^3 to 9.4315×10^3. σ_K at the same time increases from 0.1876×10^3 to 1.1702×10^3. Our event-triggered algorithm is about two order magnitude faster than the dual decomposition. We can also see that, unlike the dual decomposition algorithm, which scales superlinearly with respect to $\overline{S}$, our event-triggered algorithm on the other hand is 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 event-triggered algorithm, when $\overline{L}$ increases from 4 to 18, m_K first increases from 34 to 67 (when $\overline{L} = 11$), then varies between 65 and 68 when $\overline{L} > 11$. σ_K varies between 12 and 36. For dual decomposition, m_K increases from 1.8009×10^3 to 8.5434×10^3. σ_K at the same time increases from 0.1773×10^3 to 1.1411×10^3. Our event-triggered algorithm is about two order magnitude faster than the dual decomposition. We can also see that, unlike the dual decomposition algorithm, which scales superlinearly with respect to $\overline{L}$, our event-triggered algorithm on the other hand is scale-free.

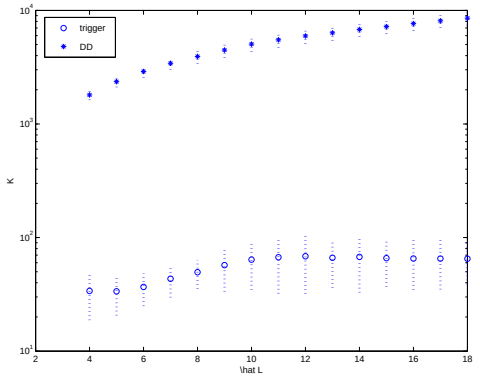

Fig. 2. Iteration number K as a function of $\overline{L}$ for both algorithms.

VI. CONCLUSION

This paper presents an event-triggered distributed NUM algorithm based on the augmented Lagrangian 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 is scale-free with respect to two measures of network size, and reduces the number of message exchanges by two orders of magnitude when compared to existing dual decomposition algorithms. Future work will focus on analyzing how different parameters affect the performance of the algorithm.

VII. APPENDIX

A. Proof of Theorem 4.1

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

$$\begin{aligned} -\dot{L}(x;\lambda,w) &= -\sum_{i=1}^{N} \frac{\partial L}{\partial x_i}\frac{dx_i}{dt} \\ &= \sum_{i=1}^{N} z_i[\nabla U_i(x_i) - \sum_{j=1}^{M} \mu_j A_{ji}] \quad (26) \\ &\geq \sum_{i=1}^{N}\left\{\frac{1}{2}z_i^2 - \frac{1}{2}[\sum_{j=1}^{M}(\mu_j-\hat{\mu}_j)A_{ji}]^2\right\} \quad (27) \end{aligned}$$

The last inequality holds whether the positive projection is active or not for each user i. Also 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

$$-\left[\sum_{j=1}^{M}(\mu_j-\hat{\mu}_j)A_{ji}\right]^2 \geq -|\mathcal{L}_i|\sum_{j=1}^{M}[(\mu_j-\hat{\mu}_j)A_{ji}]^2 \quad (28)$$

we have

$$\begin{aligned} &-\dot{L}(x;\lambda,w) \\ &\geq \frac{1}{2}\sum_{i=1}^{N} z_i^2 - \frac{1}{2}\sum_{i=1}^{N}\left\{|\mathcal{L}_i|\sum_{j=1}^{M}[(\mu_j-\hat{\mu}_j)A_{ji}]^2\right\} \quad (29) \\ &= \frac{1}{2}\sum_{i=1}^{N} z_i^2 - \frac{1}{2}\sum_{j=1}^{M}\left\{(\mu_j-\hat{\mu}_j)^2\sum_{i=1}^{N}|\mathcal{L}_i|A_{ji}^2\right\} \quad (30) \\ &\geq \frac{1}{2}\sum_{i=1}^{N} z_i^2 - \frac{1}{2}\sum_{j=1}^{M}\overline{LS}(\mu_j-\hat{\mu}_j)^2 \quad (31) \end{aligned}$$

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

$$\begin{aligned} &\frac{1}{2}\rho\sum_{i=1}^{N}\hat{z}_i^2 = \frac{1}{2}\rho\sum_{i=1}^{N}\overline{L}\frac{1}{\overline{L}}\hat{z}_i^2 \quad (32) \\ &= \frac{1}{2}\rho\sum_{j=1}^{M}\sum_{i=1}^{N}\frac{1}{\overline{L}}\hat{z}_i^2 A_{ji} + \frac{1}{2}\rho\sum_{i=1}^{N}(\overline{L}-|\mathcal{L}_i|)\frac{1}{\overline{L}}\hat{z}_i^2 \quad (33) \end{aligned}$$

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

$$\begin{aligned} -\dot{L}(x;\lambda,w) &\geq \frac{1}{2}\sum_{i=1}^{N} z_i^2 - \frac{1}{2}\rho\sum_{i=1}^{N}\hat{z}_i^2 \\ &\quad + \frac{1}{2}\rho\sum_{i=1}^{N}\hat{z}_i^2 - \frac{1}{2}\sum_{j=1}^{M}\overline{LS}(\mu_j-\hat{\mu}_j)^2 \quad (34) \\ &\geq \frac{1}{2}\sum_{i=1}^{N}[z_i^2 - \rho\hat{z}_i^2] + \\ &\quad \frac{1}{2}\sum_{j=1}^{M}\left\{\rho\sum_{i\in\mathcal{S}_j}\frac{1}{\overline{L}}\hat{z}_i^2 - \overline{LS}(\mu_j-\hat{\mu}_j)^2\right\} \quad (35) \end{aligned}$$

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 17 and 18 for all $\ell = 0, 1, 2, ..., \infty$, and any $i \in \mathcal{S}$, $j \in \mathcal{L}$, 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

$$z_i = \hat{z}_i = 0, \quad \forall i \in \mathcal{S}, \quad \mu_j = \hat{\mu}_j, \quad \forall j \in \mathcal{L} \quad (36)$$

which corresponds to $x^*(\lambda,w)$. As a result, the equilibrium $x^*(\lambda,w)$ 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 ϵ_i are switched according to the monotone decreasing sequences $\{\overline{\epsilon}_i[k]\}_{k=0}^{\infty}$ which are lower bounded by zero. This means after an finite or infinite number of switches, they will converge to their equilibria ϵ_i^* [15]. Next we will show $\epsilon_i^* = 0$ by contradiction.

Assume ϵ_i are at the equilibrium, then by the algorithm, for each link j, w_j and λ_j are also at their equilibrium. This means we now have fixed $w^*, \lambda^*, \epsilon^*$, which satisfy all the assumptions in theorem 4.1. Suppose at least one user r has a nonzero equilibrium ϵ_r^*. From theorem 4.1, we know $z_r(t)$ also asymptotically converges to zero and 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 tolerance 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 will show for each link j, M_j^L is also infinite. Define $T^{(0)} = \max_{i\in\mathcal{S}} T_i^{\epsilon}[0]$. Then on $t \in [0, T^{(0)}]$, each user $i \in \mathcal{S}$ has completed at least one switch. By algorithm 4.2, this means each link $j \in \mathcal{L}$ also has completed at least one switch. Starting from $T^{(0)}$, we can use

the same argument again. As a result, we can partition the time axis $[0,+\infty)$ into the union of time intervals $[0,T^{(0)}]\bigcup(T^{(0)},T^{(1)}]\bigcup(T^{(1)},T^{(2)}]\cdots$. 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, we can construct an infinite number of such intervals. This means M_j^L is also infinite for each $j\in\mathcal{L}$. Proof complete. ■

C. Proof of Lemma 4.3

Proof: Remember

$$\begin{aligned}
&-\dot{L}(x;\lambda,w)\\
=&\sum_{i=1}^{N} z_i[\nabla U_i(x_i)-\sum_{j=1}^{M}\mu_j A_{ji}] &(37)\\
\leq&\frac{1}{2}\sum_{i=1}^{N}\left\{z_i^2+\left[z_i+\sum_{j=1}^{M}\hat{\mu}_j A_{ji}-\sum_{j=1}^{M}\mu_j A_{ji}\right]^2\right\} &(38)\\
\leq&\frac{3}{2}\sum_{i=1}^{N}z_i^2+\sum_{i=1}^{N}\left\{\left[\sum_{j=1}^{M}\hat{\mu}_j A_{ji}-\sum_{j=1}^{M}\mu_j A_{ji}\right]^2\right\} &(39)
\end{aligned}$$

As a result of the events in theorem 4.1, the right-hand side of equation 27 in the proof is nonnegative, which is

$$\sum_{i=1}^{N}z_i^2-\sum_{i=1}^{N}\left\{\left[\sum_{j=1}^{M}\hat{\mu}_j A_{ji}-\sum_{j=1}^{M}\mu_j A_{ji}\right]^2\right\}\geq 0 \qquad (40)$$

This means

$$-\dot{L}(x;\lambda,w)\leq\frac{3}{2}\sum_{i=1}^{N}z_i^2+\sum_{i=1}^{N}z_i^2=\frac{5}{2}\sum_{i=1}^{N}z_i^2 \qquad (41)$$

which completes the proof. ■

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 nondecreasing sorted sequence of all the elements in $\{T_i^{\epsilon}[k]\}_{k=0}^{M_i^S}$ for all $i\in\mathcal{S}$. This means we can partition the time axis $[0,+\infty)$ into the union of infinite number of time intervals, $[0,T[1])\bigcup[T[1],T[2])\bigcup[T[2],T[3])\cdots$. Here $T[k]$ is the time instant when a user tolerance switch occurs for any user. On $[T[k],T[k+1])$, we have fixed parameter w,λ,ϵ. By lemma 4.3, we have

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

where $\tilde{\epsilon}_i(t)$ is defined as

$$\tilde{\epsilon}_i(t)=\overline{\epsilon}_i[k-1],\quad t\in[T_i^{\epsilon}[k],T_i^{\epsilon}[k+1]) \qquad (42)$$

Since $T_i^{\epsilon}[k]$ is the time instant when the kth tolerance switch occurs for user i. At any time t, $\tilde{\epsilon}_i(t)$ is the tolerance for user i right before the latest switch occurs.

Define $f(t)=-\frac{5}{2}\sum_{i=1}^{N}\tilde{\epsilon}_i^2(t)$, we know $f(t)$ is a nondecreasing function of t that converges to 0. Also for each user $i\in\mathcal{S}$, define $g_i(t)=\tilde{\epsilon}_i(t)$. Then $g_i(t)$ is also a nondecreasing function of t that converges to 0. This means $|z_i(t)|$ converges to zero as $t\to\infty$. Similarly, $\dot{L}(x;\lambda,w)$ also converges to zero as $t\to\infty$.

Remember equation 35 and the user and link events in algorithms 4.1-4.2, this immediately implies $\forall i\in\mathcal{S},\forall j\in\mathcal{L}$

$$\lim_{t\to\infty}\{z_i^2(t)-\rho\hat{z}_i^2(t)\}=0 \qquad (43)$$

$$\lim_{t\to\infty}\{\rho\sum_{i\in\mathcal{S}_j}\frac{1}{\overline{L}}\hat{z}_i^2(t)-\overline{LS}(\mu_j(t)-\hat{\mu}_j(t))^2\}=0 \qquad (44)$$

From equation 43 and the fact that $\lim_{t\to\infty}z_i(t)=0$, we have $\lim_{t\to\infty}\hat{z}_i(t)=0$. When combined with equation 44, we obtain $\lim_{t\to\infty}|\mu_j(t)-\hat{\mu}_j(t)|=0$. This means

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

So as $t\to\infty$, rate $x(t)$ comes closer and closer to satisfying the Karush-Kuhn-Tucker conditions of the original NUM problem. 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] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] X. Wang and M. Lemmon, "State based Self-triggered Feedback Control Systems with L2 Stability," in *Proceedings of the 17 IFAC World Congress*, 2008.

[12] ——, "Event-triggered Broadcasting across Distributed Networked Control Systems," in *Proceedings of the American Control Conference*, 2008.

[13] P. Wan and M. D. Lemmon, "Distributed Network Utility Maximization using Event-triggered Barrier Methods," *Submitted to European Control Conference 2009*.

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

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