This Dissertation

entitled

Event-triggered distributed algorithms for network optimization

typeset with NDdiss2$_\varepsilon$ v3.0 (2005/07/27) on November 30, 2009 for

Pu Wan

This LaTeX 2$_\varepsilon$ classfile conforms to the University of Notre Dame style guidelines established in Spring 2004. However it is still possible to generate a non-conformant document if the instructions in the class file documentation are not followed!

> Be sure to refer to the published Graduate School guidelines at `http://graduateschool.nd.edu` as well. Those guidelines override everything mentioned about formatting in the documentation for this NDdiss2$_\varepsilon$ class file.

It is YOUR responsibility to ensure that the Chapter titles and Table caption titles are put in CAPS LETTERS. This classfile does *NOT* do that!

This page can be disabled by specifying the "`noinfo`*" option to the class invocation.* (i.e.,`\documentclass[...,noinfo]{nddiss2e}`)

This page is *NOT* part of the dissertation/thesis, but MUST be turned in to the proofreader(s) or the reviwer(s)!

NDdiss2$_\varepsilon$ documentation can be found at these locations:

`http://www.gsu.nd.edu`
`http://graduateschool.nd.edu`

Event-triggered distributed algorithms for network optimization

A Dissertation

Submitted to the Graduate School

of the University of Notre Dame

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy in Electrical Engineering

by

Pu Wan, M.S. in Electrical Engineering

Michael D. Lemmon, Director

Graduate Program in Department of Electrical Engineering

Notre Dame, Indiana

November 2009

Event-triggered distributed algorithms for network optimization

Abstract

by

Pu Wan

Many existing distributed algorithms for network optimization problems often rely on the fact that, if the communications between subsystems are frequent enough, then the state of the network will converge to its optimum asymptotically. This approach will generally incur large communication cost. This work investigates the use of event-triggered communication schemes in distributed network optimization algorithms. Under event triggering, each subsystem broadcasts to its neighbors when a local "error" signal exceeds a state dependent threshold. We use the network utility maximization (NUM) problem as an example to demonstrate our idea.

We first present an event-triggered distributed barrier algorithm and prove its convergence. The algorithm shows significant reduction in the communication cost of the network. However, the algorithm suffers from several issues which limit its usefulness. We then propose two different event-triggered distributed NUM algorithms, the primal, and the primal-dual algorithm. Both algorithms are based on the augmented Lagrangian methods. We establish state-dependent event-triggering thresholds under which the proposed algorithms converge to the solution of NUM. For the primal-dual algorithm, we consider scenarios when the network has data dropouts or transmission delay, and give an upper bound on the largest number of successive data dropouts and the maximum allowable transmission delay, while ensuring the asymptotic convergence of the algorithm. A state-dependent lower bound on the broadcast period is also given. Simulations show that all proposed algorithms reduce the number of message exchanges by up to two orders of magnitude when compared to existing dual decomposition algorithms, and are scale-free with respect to two measures of network size.

We then use the optimal power flow (OPF) problem in microgrids as a nontrivial real-life example to demonstrate the effectiveness of event-triggered optimization algorithms. We develop an event-triggered distributed algorithm for the OPF problem and prove its convergence. We use the CERTS microgrid model as an example power system to show the effectiveness of our algorithm. The simulation is done in MATLAB/SimPower and shows that our algorithm solves the OPF problem in a distributed way, and the communication between neighboring subsystems is very infrequent.

CONTENTS

FIGURES

TABLES

CHAPTER 1

Introduction

1.1 Motivation

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.

Many problems in such networks can be formulated as optimization problems. This includes estimation [44] [47] [25], source localization [44], data gathering [13] [12], routing [35], control [52], resource allocation [43] [65] in sensor networks, resource allocation in wireless communication networks [64] [14], congestion control in wired communication networks [27] [34], and optimal power dispatch [29] in electrical power grid. The consensus problem [39] can also be viewed as a distributed optimization problem where the objective function is the total state mismatch between neighboring agents. Due to the large-scale nature and complexity of the system, centralized optimization techniques need an unacceptably large amount of coordination and signaling. When distributed optimization algorithms are used, each subsystem communicates with a collection of other subsystems, and all subsystems collaboratively solve the overall optimization problem.

Early distributed algorithms that solve the network optimization problem directly include [23] [49]. Their results suggest that if the communication between subsystems is frequent enough, then the state of network will converge to its optimal point asymptotically. Recently, several other distributed optimization algorithms have been proposed. In [44], a parameter estimation problem for sensor networks was proposed. A parameter estimate is circulated through the network, and along the way each node makes a small gradient descent-like adjustment to the estimate based only on its local data. It was shown that the distributed algorithm converges to an approximate solution for a broad class of estimation problems. In [25], a randomized incremental subgradient method was used, where each node only needs to exchange information with its neighbors. This distributed algorithm is also shown to converge to a small neighborhood of the optimal solution.

Recent related developments in distributed algorithm has been largely in the networking community. Most algorithms focused on solving the Network Utility Maximization (NUM) problem. NUM problems maximize a global separable measure of the networked system's performance subject to linear inequality constraints. It originates in the Internet context [27] [34] as a way to understand Internet protocols such as TCP. The NUM problem has a rather general form and many problems in other areas can be formulated as a NUM problem with little or no variation. As a matter of fact, all the problem we mentioned previously can be formulated as a NUM problem with little or no variation. It is for this reason that we will use the general NUM problem formulation as an illustrative example to demonstrate the idea of our event-triggered algorithm. However, we must emphasize that our objective is not trying to solve the Internet congestion control problem in [34] using a better algorithm.

A variety of distributed algorithms have been proposed to solve the NUM problem [27] [34] [63] [42]. We will explain the main results on distributed algorithms for the NUM problem in detail later in section 1.3. Kelly [27] first proposed two classes of algorithms by decomposing the NUM problem into a user problem and a network problem. In these algorithms, the network sets a "shadow-price" for the user's consumption of network resources. The user must then balance the utility it receives by consuming such resources against the actual cost of the resources. Simultaneously, the network adjusts its price to maximize its total revenue. The interplay between user consumption and shadow pricing forms a feedback system in which the users and network play a cooperative game. Kelly showed that this game could be implemented in a distributed manner in which the locally greedy actions of resources and users converge to the solution of the global optimization problem.

The NUM 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. [34] 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 (as measured by the number of iterations) that scales in a super-linear manner with those two measures of network size, $\overline{L}$ and $\overline{S}$. Similar issues exist in earlier algorithms [23] [49] [25], where the communications between subsystems are assumed to be "frequent enough", which is equal to choosing a small underlying step size in subsystem state update.

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 [21]. 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.

Our work presents one way of reducing the message passing complexity of distributed optimization algorithms. It has recently been demonstrated [61] that event-triggering in state feedback control systems can greatly lengthen the average sampling period of such systems. The result suggests that the use of event-triggering in a suitable distributed algorithm may significantly reduce the message passing complexity experienced by such algorithms. Event-triggering eliminates the need for using a pre-selected constant step size in the distributed algorithm. Under event triggering each subsystem broadcasts its state information when some local 'error' signal exceeds a certain threshold.

The rest of the chapter reviews the major prior work that is related to our research. In section 1.2, the Network Utility Maximization (NUM) problem is formally stated. Section 1.3 reviews existing distributed algorithms for solving the NUM problem. Three of the most representative papers are discussed in more detail. Section 1.4 reviews some results in event-triggered control. The outline of the thesis is then presented in section 1.5.

1.2 The Network Utility Maximization (NUM) problem

The network utility maximization (NUM) problem [27] [34] considers a network consisting of N users and M 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 problem generally takes the form

$$\begin{aligned} \text{maximize:} \quad & U(x) = \textstyle\sum_{i=1}^{N} U_i(x_i) \\ \text{subject to:} \quad & Ax \leq c \quad x \geq 0 \end{aligned} \tag{1.1}$$

where $x = [x_1, ..., x_N]^T$, $x_i \in \mathbb{R}$ is user i's data rate, $A \in \mathbb{R}^{M \times N}$ represents the routing matrix, and $c \in \mathbb{R}^M$ is the capacity vector of the M links. $U_i(x_i)$ is the utility function of user i, which has the following properties:

1. $U_i(x_i)$ is twice differentiable and strictly increasing, i.e. $\nabla U_i(x_i) > 0$.

2. $U_i(x_i)$ is strictly concave in x_i, i.e. $\nabla^2 U_i(x_i) < 0$.

The utility function $U_i(x_i)$ usually represents the reward (i.e. quality-of-service) [27][34] user i gets by transmitting at rate x_i.

For the routing matrix A, if $A_{ji} = 1$, link j is used by user i; if $A_{ji} = 0$, then link j is not used by user i. The jth row of Ax represents the total data rates that go through link j, which cannot exceed its capacity c_j. The constraint c usually represents the bandwidth capacity of the link. For a large scale network, A

usually has sparse structure. People are interested in using a distributed algorithm to solve the NUM problem. The term ‘distributed’ means each user only knows the information of the links that it uses, and each link only knows the information of the users that use it. Figure 1.1 is an example network with the following constraint:

$$Ax = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \leq \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

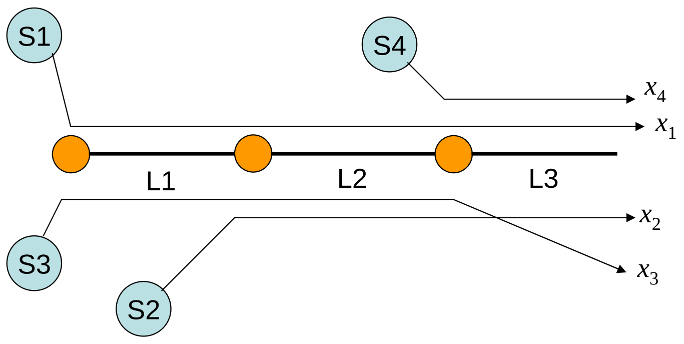

Figure 1.1. An example network in NUM

The solution to the NUM problem maximizes the summed utility seen by all users in the network as a function of the users’ transmission rates. These rates must clearly be non-negative. Moreover, if $U_i(x) = w_i \log x$ where w_i is a positive constant, then it can be shown that all of the user rates in the optimal solution must be positive. Such solutions are seen as being “fair”, since no user's optimal rate goes to zero.

We use $\mathcal{S} = \{1, ..., N\}$ to denote the set of users, and $\mathcal{L} = \{1, ..., M\}$ to denote the set of links. For user i, $\mathcal{L}_i = \{j \in \mathcal{L} | i \in \mathcal{S}_j\}$ is the set of links used by it. For link j, $\mathcal{S}_j = \{i \in \mathcal{S} | j \in \mathcal{L}_i\}$ is the set of its users. Note that $j \in \mathcal{L}_i$ if and only if $i \in \mathcal{S}_j$. We also define $\overline{L} = \max_{i \in \mathcal{S}} |\mathcal{L}_i|$, $\overline{S} = \max_{j \in \mathcal{L}} |\mathcal{S}_j|$. In other words, $\overline{L}$ is the maximum number of links any user $i \in \mathcal{S}$ uses, and $\overline{S}$ is the maximum number of users any link $j \in \mathcal{L}$ has.

1.3 Prior work: Distributed algorithms for NUM

This section reviews existing distributed algorithms used in solving the NUM problem. The NUM 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. In the algorithms given in [27][34], either the user or the link update is a memoryless penalty function, which corresponds to a dual or primal algorithm, respectively. In [41][6], a dual algorithm is used, and the link update is implemented by a second order dynamical equation. The cases when both user and link updates are dynamic have been analyzed in [31][2][24]. In [31], stability was shown using singular perturbations, while in [2][24], stability was established only for the single-bottleneck case (i.e., only one congested link is considered). In [63], the author used passivity to establish the stability of a broad class of primal-dual algorithms, and included the primal and dual algorithms of [27][41] as special cases.

Among the existing literature, the NUM problem is mostly solved using a dual algorithm. The approach was first applied to the NUM problem in [27], and then later widely used in the Internet [34] [15], wireless ad hoc networks [43] [65] [66], and sensor networks [12].

Next we will review three of the most representative papers in more detail. Initial work on a distributed solution to the NUM problem is presented in subsection 1.3.1. Subsection 1.3.2 discusses the dual decomposition approach. Passivity-based primal-dual algorithms are presented in subsection 1.3.3.

1.3.1 Kelly's Initial Work

Kelly [27] first proposed two classes of algorithms to solve the NUM problem (equation 1.1) and proved the convergence of these algorithms by constructing a Lyapunov function. The utility function for user $i \in \mathcal{S}$ was chosen explicitly as a logarithm function $U_i(x_i) = w_i \log x_i$, where $w_i > 0$ is a weighting coefficient. For this utility function, the primal algorithm takes the form

$$\begin{aligned} \dot{x}_i(t) &= \gamma \left(w_i - x_i(t) \sum_{j \in \mathcal{L}_i} p_j(t) \right) & (1.2) \\ p_j(t) &= h_j \left(\sum_{i \in \mathcal{S}_j} x_i(t) \right) & (1.3) \end{aligned}$$

where $\gamma > 0$ is some gain constant. $p_j(t)$ is link j's feedback signal, which has the interpretation of the price for using the link. $h_j(\cdot)$ is a continuous, non-negative, increasing function which takes the form

$$h_j(y) = [y - c_j + \varepsilon]^+ / \varepsilon^2 \tag{1.4}$$

where $[z]^+ = \max\{z, 0\}$ and ε is a small positive constant. Equations 1.2-1.3 are called the primal algorithm because the user's data rate x (which is the primal variable) is updated using a first order differential equation. On the other hand, the link's update is a memoryless function of the user rate. In equation 1.2, $w_i/x_i(t)$ can be interpreted as user i's willingness-to-pay when it transmits at rate $x_i(t)$. Then according to equation 1.2, user i increases its rate if the total price $\sum_{j\in\mathcal{L}_i} p_j(t)$ is less than its willingness-to-pay, and decreases it otherwise. In equation 1.3, since $h_j(\cdot)$ is an increasing function, the price $p_j(t)$ increases if the aggregate rates in link j increases.

It was shown that

$$V(x) = \sum_{i\in\mathcal{S}} U_i(x_i) - \sum_{j\in\mathcal{L}} \int_0^{\sum_{i\in\mathcal{S}_j} x_i(t)} h_j(y)dy \tag{1.5}$$

is a Lyapunov function for the system in equations 1.2-1.3. Therefore the system has an equilibrium point that is globally asymptotically stable. As $\varepsilon \to 0$, the equilibrium of equations 1.2-1.3 approaches arbitrarily close to the optimal solution of the NUM problem. If we view $V(x)$ as the penalty function of the NUM problem, then equations 1.2-1.3 can be thought of as a gradient descent algorithm to solve the problem [15].

The dual algorithm proposed by Kelly takes the form

$$\begin{aligned} \dot{p}_j(t) &= \gamma\left(\sum_{i\in\mathcal{S}_j} x_i(t) - q_j(p_j(t))\right) && (1.6)\\ x_i(t) &= \frac{w_i}{\sum_{j\in\mathcal{L}_i} p_j(t)} && (1.7) \end{aligned}$$

where $q_j(\cdot)$ is

$$q_j(\eta) = \frac{c_j\eta}{\eta+\varepsilon} \tag{1.8}$$

for small positive constant ε. In the dual algorithm 1.6-1.7, the user update is static and the link update is dynamic. Similarly, the function

$$V(x) = \sum_{i\in\mathcal{S}} U_i\left(\sum_{j\in\mathcal{L}_i} p_j(t)\right) - \sum_{j\in\mathcal{L}} \int_0^{p_j(t)} q_j(\eta)d\eta \tag{1.9}$$

is a Lyapunov function for the system in equations 1.6-1.7. This means the system's equilibrium point is globally asymptotically stable. Again as $\epsilon \to 0$, this equilibrium point approaches the solution of the NUM problem.

We should point out that Kelly in [27] relaxes the nonnegativity constraint on user rates x and link prices p. This simplifies the stability proof using Lyapunov techniques.

1.3.2 Dual decomposition approach

Dual decomposition [34] works by considering the dual problem of the NUM problem, which is given as

$$\begin{aligned} \text{minimize:} \quad & D(p) \\ \text{subject to:} \quad & p \geq 0 \end{aligned} \tag{1.10}$$

where

$$D(p) \;=\; \max_{x\geq 0} L(x,p) = \max_{x\geq 0}\{\sum_{i=1}^{N} U_i(x_i) - p^T(Ax-c)\} \tag{1.11}$$

$p = [p_1, ..., p_M]^T$ is the Lagrange multiplier (which has the interpretation of price for using each link [27] [34]) 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. [34] 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

$$x_i[k+1] = \arg\max_{x_i \geq 0} \left\{ U_i(x_i[k]) - x_i[k] \sum_{j \in \mathcal{L}_i} p_j[k]) \right\} \tag{1.12}$$

$$p_j[k+1] = \max \left\{ 0, p_j[k] + \gamma \left\{ \sum_{i \in \mathcal{S}_j} x_i[k] - c_j \right\} \right\} \tag{1.13}$$

for $k = 0, \cdots, \infty$. $x_i[k]$ and $p_j[k]$ will converge to the optimal value x^* and p^*, respectively. In equation 1.12, user i only needs to know the aggregate prices of all the links that user i uses at time k, which is known to user i. In the link update law in equation 1.13, link j only needs to know the total data rates that pass through itself, which is also known to link j. The above remarks show that dual decomposition yields a fully distributed computational solution of the NUM problem.

The update law in equations 1.12-1.13 is very similar to the dual algorithm 1.6-1.7 in Kelly's work. In the special case where $U_i(x_i) = w_i \log x_i$, equations 1.12-1.13 reduces to a similar form of 1.6-1.7, provided $q_j(p_j(t)) = c_j$ in equation 1.6. However, this choice of $q_j(\cdot)$ does not satisfy certain conditions required for the stability proof in [27].

The stability of the system in equations 1.12-1.13 is proved by showing that the dual objective function $D(p(t))$ can be thought of as a Lyapunov function for the discrete time system, provided the step size γ is sufficiently small. It was shown in [34] that the stabilizing γ satisfies

$$0 < \gamma < \gamma^* = \frac{-2 \max_{(i,x_i)} \nabla^2 U_i(x_i)}{\overline{L}\overline{S}} \tag{1.14}$$

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 1.14 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 $\overline{S}$.

If the network is highly congested (large $\overline{S}$), or has a long route (large $\overline{L}$), then from equation 1.14, we have to choose small γ to ensure stability [5]. When the utility function is in the form of a logarithm function $U_i(x_i) = w_i \log x_i$, then the optimal step size is

$$\gamma^* = \frac{2}{\overline{L}\overline{S}} \min_{i \in \mathcal{S}} (\frac{w_i}{x_i^2}), \tag{1.15}$$

This means that the step size will depend on the maximum data rate the users can transmit.

In real Internet congestion control protocols, the price p is usually implicitly fed back to the users. Various protocols have been developed, such as RED[19], REM[6] and RAM[1]. Such mechanism is certainly more practical than explicit transmission of price information and suffices in the Internet context. However, it suffers from various errors inherent in its implicit price-notification [36], which limits its applicability. Moreover, in many applications, such as in the consensus[39] and electrical power grid [29] problem, there is no way we can employ this kind of implicit pricing feedback scheme. For this reason, explicit feedback scheme is still needed in many applications. Our later presented event-triggered algorithm uses an explicit feedback scheme. Since we are interested in solving a large class of optimization problems instead of just the simple NUM problem in the Internet context, the scheme serves our purposes well.

1.3.3 Passivity Approach

Consider the following primal-dual algorithm, where both the user update and the link update are dynamic,

$$\begin{aligned} \dot{x}_i(t) &= k_i \left(\nabla U_i(x_i) - \sum_{j \in \mathcal{L}_i} p_j(t) \right)^+_{x_i(t)} && (1.16) \\ \dot{p}_j(t) &= \gamma_j \left(\sum_{i \in \mathcal{S}_j} x_i(t) - c_j \right)^+_{p_j(t)} && (1.17) \end{aligned}$$

Given a function $f(x)$, its positive projection is defined as

$$(f(x))^+_x = \begin{cases} f(x), & \text{if} \quad x > 0, \\ f(x), & \text{if} \quad x = 0 \quad \text{and} \quad f(x) \geq 0 \\ 0, & \text{if} \quad x = 0 \quad \text{and} \quad f(x) < 0 \end{cases} \tag{1.18}$$

If we denote $y = Ax$, then $y_j(t) = \sum_{i \in \mathcal{S}(j)} x_i(t)$. Denote the optimal value of y as y^*, and the optimal value of p as p^*. It was shown in [63] that for the user algorithm 1.16, the system from $-(p - p^*)$ to $y - y^*$ is strictly passive if we choose the storage function

$$V_1(x) = \frac{1}{2} \sum_{i \in \mathcal{S}} \frac{(x_i - x_i^*)^2}{k_i} \tag{1.19}$$

Similarly, for the link algorithm 1.17, the system from $y - y^*$ to $p - p^*$ is strictly passive if we choose the storage function

$$V_2(p) = \frac{1}{2} \sum_{j \in \mathcal{L}} \frac{(p_j - p_j^*)^2}{\gamma_j} \tag{1.20}$$

By the passivity theorem [28, Chap 6], the feedback interconnection of 1.16 and 1.17 is asymptotically stable, and $V(x,p) = V_1(x) + V_2(p)$ can be used as a Lyapunov function for the interconnected system. This allows us to use a primal-dual algorithm, where both the user and link update are dynamic. We should emphasize here that the classes of algorithms are not restricted to equations 1.16-1.17 only, any algorithms satisfying certain passivity properties can be used here.

As we can see, existing solutions to the NUM problem either rely on conservative choice of step size in order to ensure stability, or otherwise stated as a continuous-time law, which makes it difficult for us to determine how often the communication is needed. This inspires us to use an event-triggered scheme which eliminates the need for pre-selected constant step size in the distributed algorithm.

1.4 Prior work: Event-triggered control

Event-triggered systems are systems in which sensors are sampled in a *sporadic* non-periodic manner. Event-triggering has an agent transmit information to its neighbors when some measure of the "novelty" in that information exceeds some specified threshold. In this way, the communication network is only used when there is an immediate need. Early examples of event-triggering were used in relay control systems[50] and more recent work has looked at event-triggered PID controllers [3]. Much of the early work in event-triggered control assumed event-triggers in which the triggering thresholds were constant. Recently it was shown that state-dependent event triggers could be used to enforce stability concepts such as input-to-state stability [48] or $\mathcal{L}_2$ stability [60] in both embedded control systems and networked control systems [62]. There has been ample experimental evidence [4, 46] to suggest that event-triggering can greatly reduce communication bandwidth while preserving overall system performance. Event-triggering therefore provides a useful approach for reducing an application's usage of the communication network.

1.5 Outline of thesis

The outline of the thesis is briefly introduced in the following paragraphs, sorted by chapter.

Chapter 2 Existing distributed optimization algorithms usually consume a large amount of communication resources because of the inherent coupling between inter-subsystem communication and local computation. This chapter addresses this issue through the use of an event-triggered scheme. In this chapter we start to study how to use event-triggering to separate communication and computation in distributed optimization algorithms.

The chapter uses the NUM problem formulation as an example and presents a distributed algorithm based on barrier methods that uses event-triggered message passing. Under event triggering, each subsystem broadcasts to its neighbors when a local "error" signal exceeds a state dependent threshold. We prove that the proposed algorithm converges to the global 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 when compared to dual decomposition algorithms, and is scale-free with respect to two measures of network size $\overline{L}$ and $\overline{S}$.

Chapter 3 The event-triggered barrier algorithm in chapter 2 is our very first approach to solve the distributed optimization problems using the event-triggered idea, and it shows significantly reduction in the communication cost of the network. However, as we show in the chapter, the algorithm suffers from issues like the need for an initial feasible point, and performance sensitive to parameter choices. All these issues limit the usefulness the algorithm. This inspired us to look for alternative algorithms to apply the event-triggered idea.

This chapter still uses the NUM problem formulation as an example and presents two distributed algorithms based on the augmented Lagrangian methods that use event-triggered message passing and proves their convergence. One of the algorithm is a primal algorithm, and the other is a primal-dual algorithm. For the primal-dual algorithm, we consider scenarios when the network has data dropouts, and give an upper bound on the largest number of successive data dropouts each link can have, while ensuring the asymptotic convergence of the algorithm. A state-dependent lower bound on the broadcast period and an upper bound on the transmission delay the network can tolerate while ensuring convergence are also given. Simulation results show that both algorithms have a message passing complexity that is up to two orders of magnitude lower than dual decomposition algorithms, and are scale-free with respect to two measures of network size $\overline{L}$ and $\overline{S}$. The primal algorithm in this chapter is similar to the algorithm in chapter

2. However, in chapter 2, 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, need for an initial feasible point and sensitive to the choice of parameters. The event-triggered algorithms presented in this chapter do not have these issues.

Chapter 4 The problems and algorithms developed in the previous chapters are somewhat abstract, and did not explicitly show how we can use those event-triggered algorithms in real-life applications. This chapter uses the optimal power flow problem in microgrids as a nontrivial real-life example to demonstrate the effectiveness of event-triggered optimization algorithms.

The optimal power flow (OPF) problem has been the research subject of power system community since early 1960's, and is very similar to the NUM problem we considered in previous chapters. Various centralized or distributed optimization algorithms have been proposed to solve the OPF problem. These algorithms usually made the assumptions that communication between subsystems was not expensive and reliable, which is unrealistic. One way around this problem is to make use of low power ad hoc wireless communication networks that operate independently of the main power grid. Using event-triggering on those wireless networks therefore may provide a useful approach for reducing the power grid's use of the communication network.

In this chapter, we develop an event-triggered distributed optimization algorithm for the OPF problem and prove its convergence. We use the CERTS microgrid model [32] as an example power system to show the effectiveness of our algorithm. The simulation is done in MATLAB/SimPower and shows that our algorithm solves the OPF problem in a distributed way, and the communication between neighboring subsystems is very infrequent.

Chapter 5 This chapter is a summary of the thesis, and also provides suggestions for future work.

CHAPTER 2

Event-triggered distributed optimization using barrier methods

2.1 Introduction

This chapter presents one way of reducing the message passing complexity of distributed optimization algorithms using event-triggering. The chapter uses the NUM problem formulation as an example and presents a distributed algorithm based on barrier methods that uses event-triggered message passing. Under event triggering, each subsystem broadcasts to its neighbors when a local "error" signal exceeds a state dependent threshold. We prove that the proposed algorithm converges to the global 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 two measures of network size $\overline{L}$ and $\overline{S}$.

The rest of the chapter is organized as follows. The event-triggered optimization algorithm is based on a barrier method solution to the NUM problem which is described in section 2.2. Section 2.3 presents our event-triggered distributed algorithm based on barrier methods, and proves its convergence. Simulation results are shown in section 2.4, and section 2.5 concludes the chapter.

2.2 Barrier Method NUM Algorithm

The event-triggered algorithm presented in this chapter 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 [67]. 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. The solutions to the sequence of unconstrained problems form the central path, which is a 1-D curve connecting the initial point in the feasible region to the optimal point.

Traditional barrier algorithms [38] 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 the NUM problem is

$$L(x;\tau,\lambda) = -\sum_{i\in\mathcal{S}} U_i(x_i) - \sum_{i\in\mathcal{S}} \lambda_i \log x_i - \sum_{j\in\mathcal{L}} \tau_j \log(c_j - a_j^T x) \tag{2.1}$$

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.

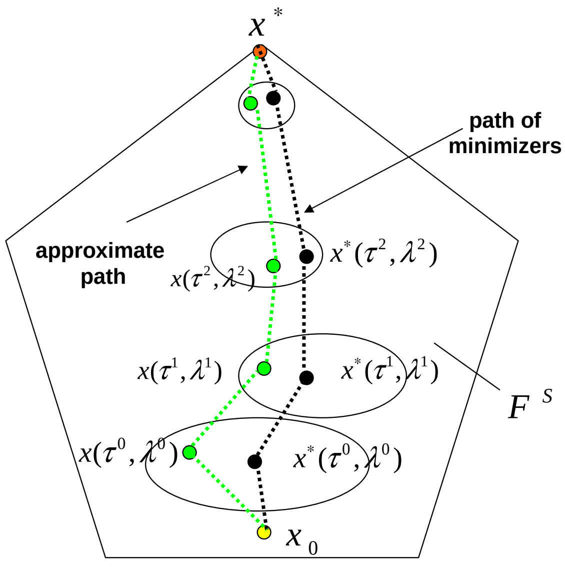

Figure 2.1. Illustration of the barrier methods

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 [38], the sequence of approximate minimizers $\{x^*[k]\}_{k=0}^{\infty}$ converges to the optimal point of the NUM problem, which is illustrated in figure 2.1. The barrier method algorithm for the NUM problem is formally stated below.

1. **Initialization:** Select an initial user rate $x^0 \in \mathcal{F}^s$ and set $K = 0$. Set $\tau_j = \overline{\tau}_j[K]$, $\lambda_i = \overline{\lambda}_i[K]$, $i \in \mathcal{S}$, $j \in \mathcal{L}$, and $\epsilon = \overline{\epsilon}[K]$.

2. **Main Recursive Loop:**

   Do until: $\|\nabla_x L(x^0, \tau, \lambda)\| \leq \epsilon_d$

   (a) **Approximately Minimize $L(x; \tau, \lambda)$:**

   Do until: $\|\nabla_x L(x^0; \tau, \lambda)\| \leq \epsilon$

$$\begin{aligned} x &= x^0 - \gamma \nabla_x L(x^0; \tau, \lambda) \\ x^0 &= x \end{aligned} \tag{2.2}$$

   (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. 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 2.3.

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 between links and users is equal to the number of updates required for the algorithm's convergence. That number is, of course, 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. The following section presents an event-triggered implementation of the barrier method NUM algorithm. It reduces the communication cost of the algorithm significantly.

2.3 Event-triggered NUM Barrier Algorithm

The NUM barrier 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 $L(x; \overline{\tau}[k], \overline{\lambda}[k])$ where $\{\overline{\tau}[k]\}_{k=0}^{\infty}$ and $\{\overline{\lambda}[k]\}_{k=0}^{\infty}$ are sequences of link and user barrier vectors, respectively, that are monotone decreasing to zero.

Implementing the NUM barrier algorithm in a distributed manner requires communication between users and links. An event-triggered implementation of the NUM barrier 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 barrier 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 2.3.1 determines an event threshold condition ensuring the convergence of the local update (equation 2.2) to this minimizer. Subsection 2.3.2 then considers the case when the barrier vectors are *switched* as we change k. In particular, we present a distributed update strategy for the barrier parameters that ensures the convergence of the algorithm to the NUM problem's solution.

2.3.1 Fixed Barrier Parameter case

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

$$\begin{aligned} x_i(t) &= -\int_0^t \nabla_{x_i} L(x(s);\tau,\lambda)ds \\ &= \int_0^t \left(\frac{\partial U_i(x_i(s))}{\partial x_i} + \frac{\lambda_i}{x_i(s)} - \sum_{j\in\mathcal{L}_i} \mu_j(s) \right) ds \end{aligned} \tag{2.3}$$

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

$$\mu_j(t) = \frac{\tau_j}{c_j - a_j^T x(t)} \tag{2.4}$$

Equation 2.3 is the continuous-time version of the update in equation 2.2. Note that in equation 2.3, 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 2.4, 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 2.3. Rather than assuming each user can access the link state μ_j in a continuous fashion, 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

$$\hat{\mu}_j(t) = \mu_j(T_j^L[\ell]) \tag{2.5}$$

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 2.3 takes the form

$$x_i(t) \quad = \quad \int_0^t \left(\frac{\partial U_i(x_i(s))}{\partial x_i} + \frac{\lambda_i}{x_i(s)} - \sum_{j \in \mathcal{L}_i} \hat{\mu}_j(s) \right) ds \tag{2.6}$$

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

$$\begin{aligned} z_i(t) &= \dot{x}_i(t) \\ &= \nabla U_i(x_i(t)) + \frac{\lambda_i}{x_i(t)} - \sum_{j \in \mathcal{L}_i} \hat{\mu}_j(t) \end{aligned} \tag{2.7}$$

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

$$\hat{z}_i(t) = z_i(T_i^S[\ell]) \tag{2.8}$$

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.

We are now in a position to state the main theorem of this subsection.

Theorem 2.3.1 *Consider the Lagrangian in equation 2.1 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 barrier vector $\lambda > 0$ and $\tau > 0$ and assume the initial user rates $x_i(0)$ lie in the feasible set $\mathcal{F}^s$. 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 2.6 with sampled link states given by equation 2.5. For each $i \in \mathcal{S}$ let the user state $z_i(t)$ satisfy equation 2.7 and assume link j's measurement of the user state satisfies equation 2.8.*

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

$$z_i^2(t) - \rho \hat{z}_i^2(t) \geq 0 \tag{2.9}$$

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

$$\rho \sum_{i \in \mathcal{S}_j} \frac{1}{\overline{L}} \hat{z}_i^2(t) - \overline{LS}\,(\mu_j(t) - \hat{\mu}_j(t))^2 \geq 0 \tag{2.10}$$

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

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

$$\begin{aligned} -\dot{L}(x;\tau,\lambda) &= -\sum_{i=1}^{N} \frac{\partial L}{\partial x_i} \frac{dx_i}{dt} && (2.11) \\ &= \sum_{i=1}^{N} z_i [\nabla U_i(x_i) + \frac{\lambda_i}{x_i} - \sum_{j=1}^{M} \mu_j A_{ji}] && (2.12) \\ &\geq \sum_{i=1}^{N} \left\{ \frac{1}{2} z_i^2 - \frac{1}{2} [\sum_{j=1}^{M} (\mu_j - \hat{\mu}_j) A_{ji}]^2 \right\} && (2.13) \end{aligned}$$

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

$$-\left[\sum_{j=1}^{M} (\mu_j - \hat{\mu}_j) A_{ji} \right]^2 \geq -|\mathcal{L}_i| \sum_{j=1}^{M} [(\mu_j - \hat{\mu}_j) A_{ji}]^2 \tag{2.14}$$

we have

$$\begin{aligned}
& -\dot{L}(x;\tau,\lambda) && (2.15)\\
\geq\ & \frac{1}{2}\sum_{i=1}^{N} z_i^2 - \frac{1}{2}\sum_{i=1}^{N}\left\{|\mathcal{L}_i|\sum_{j=1}^{M}[(\mu_j-\hat{\mu}_j)A_{ji}]^2\right\} && (2.16)\\
=\ & \frac{1}{2}\sum_{i=1}^{N} z_i^2 - \frac{1}{2}\sum_{j=1}^{M}\left\{(\mu_j-\hat{\mu}_j)^2\sum_{i=1}^{N}|\mathcal{L}_i|A_{ji}^2\right\} && (2.17)\\
\geq\ & \frac{1}{2}\sum_{i=1}^{N} z_i^2 - \frac{1}{2}\sum_{j=1}^{M}\overline{LS}(\mu_j-\hat{\mu}_j)^2 && (2.18)
\end{aligned}$$

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

$$\begin{aligned}
& \frac{1}{2}\rho\sum_{i=1}^{N}\hat{z}_i^2 && (2.19)\\
=\ & \frac{1}{2}\rho\sum_{i=1}^{N}\overline{L}\frac{1}{\overline{L}}\hat{z}_i^2 && (2.20)\\
=\ & \frac{1}{2}\rho\sum_{j=1}^{M}\sum_{i=1}^{N}\frac{1}{\overline{L}}\hat{z}_i^2A_{ji} + \frac{1}{2}\rho\sum_{i=1}^{N}(\overline{L}-|\mathcal{L}_i|)\frac{1}{\overline{L}}\hat{z}_i^2 && (2.21)
\end{aligned}$$

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

$$\begin{aligned}
& -\dot{L}(x;\tau,\lambda) && \\
\geq\ & \frac{1}{2}\sum_{i=1}^{N} z_i^2 - \frac{1}{2}\rho\sum_{i=1}^{N}\hat{z}_i^2 + \frac{1}{2}\rho\sum_{i=1}^{N}\hat{z}_i^2 - \frac{1}{2}\sum_{j=1}^{M}\overline{LS}(\mu_j-\hat{\mu}_j)^2 && (2.22)\\
\geq\ & \frac{1}{2}\sum_{i=1}^{N}[z_i^2-\rho\hat{z}_i^2] + \frac{1}{2}\sum_{j=1}^{M}\left\{\rho\sum_{i=1}^{N}\frac{1}{\overline{L}}\hat{z}_i^2A_{ji} - \overline{LS}(\mu_j-\hat{\mu}_j)^2\right\} && (2.23)\\
\geq\ & \frac{1}{2}\sum_{i=1}^{N}[z_i^2-\rho\hat{z}_i^2] + \frac{1}{2}\sum_{j=1}^{M}\left\{\rho\sum_{i\in\mathcal{S}_j}\frac{1}{\overline{L}}\hat{z}_i^2 - \overline{LS}(\mu_j-\hat{\mu}_j)^2\right\} && (2.24)
\end{aligned}$$

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 2.9 and 2.10 for all $\ell = 0, 1, 2, ..., \infty$, and any $i \in \mathcal{S}$, $j \in \mathcal{L}$, then $\dot{L}(x; \tau, \lambda) \leq 0$ is guaranteed for all t.

By using the properties of $U_i(x_i)$, it is easy to show that for any fixed λ and τ, $L(x; \tau, \lambda)$ has a unique minimizer. Suppose $x^*(\tau, \lambda)$ is this minimizer, and the corresponding Lagrangian is $L(x^*; \tau, \lambda)$. 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 $\dot{L}(x; \tau, \lambda) = 0$. The only scenario this can happen is

$$z_i = \hat{z}_i = 0, \quad \forall i \in \mathcal{S}, \quad \mu_j = \hat{\mu}_j, \quad \forall j \in \mathcal{L} \tag{2.25}$$

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

Theorem 2.3.1 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 2.9 and 2.10 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 $t = T_i^S[\ell]$, we see that $z_i^2 - \rho \hat{z}_i^2$ is nonnegative so that the inequality in equation 2.9 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.

In a similar way, 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 2.9-2.10 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.

2.3.2 The switching barrier parameter case

Recall that the solution to the fixed-barrier case finds an approximate minimizer to the Lagrangian $L(x;\tau,\lambda)$ in step 2a of the NUM barrier algorithm. We now consider what happens when we systematically reduce the barrier parameters to zero. In particular, we need to identify a distributed strategy for updating these barrier 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.

The discussion of the algorithms needs an additional notation. For a function $f(t)$ defined on $t \in [0,T)$, we denote $f^+(T)$ as the limit value 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 user barrier parameters, $\{\overline{\lambda}_i[k]\}_{k=0}^{\infty}$, that asymptotically approaches zero. For each user i, we also assume there exists another monotone decreasing sequence of tolerance levels, $\{\overline{\epsilon}_i[k]\}_{k=0}^{\infty}$ that asymptotically approaches zero.

Algorithm 2.3.1 User i's Update Algorithm

1. **Parameter Initialization:** *Set the initial user rate* x_i^0 *so that* x^0 *lies in the feasible set* $\mathcal{F}^s$. *Let* $K = 0$, $T = 0$, $\lambda_i = \overline{\lambda}_i[K]$, *and* $\epsilon_i = \overline{\epsilon}_i[K]$.

2. **State Initialization:** *Upon receiving link state* $\mu_j(T)$ *from* $j \in \mathcal{L}_i$, *set* $\hat{\mu}_j = \mu_j(T)$. *Initialize user state to*

$$z_i(T) = \nabla U_i(x_i^0) + \frac{\lambda_i}{x_i^0} - \sum_{j \in \mathcal{L}_i} \hat{\mu}_j \tag{2.26}$$

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*

$$x_i(t) = \int_T^t z_i(s)ds \tag{2.27}$$

$$z_i(t) = \nabla U_i(x_i(t)) + \frac{\lambda_i}{x_i(t)} - \sum_{j \in \mathcal{L}_i} \hat{\mu}_j \tag{2.28}$$

$$x_i(T) = x_i^0 \tag{2.29}$$

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 $j \in \mathcal{L}$. 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 barrier parameters, $\{\overline{\tau}_j[k]\}_{k=0}^{\infty}$ that asymptotically approaches zero.

Algorithm 2.3.2 Link j's Update Algorithm

1. **Parameter Initialization:** *Set $K = 0$, $T = 0$, $\tau_j = \overline{\tau}_j[K]$ and set the switching indicator $I_i = 0$ for each $i \in \mathcal{S}_j$.*

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

$$\mu_j(T) = \frac{\tau_j}{c_j - \sum_{i \in \mathcal{S}_j} x_i(T)} \tag{2.30}$$

   Transmit $\mu_j(T)$ to all users $i \in \mathcal{S}_j$ and set $\hat{\mu}_j = \mu_j(T)$. Upon receiving user state $z_i(T)$ from $i \in \mathcal{S}_j$, 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 instants in time. 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 actually occurs.

Lemma 2.3.2 *Consider algorithms 2.3.1 and 2.3.2. 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.*

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 the monotone decreasing sequences $\{\overline{\lambda}_i[k]\}_{k=0}^{\infty}$ and $\{\overline{\epsilon}_i[k]\}_{k=0}^{\infty}$ which are lower bounded by zero. This means after an finite or infinite number of switches, they will converge to their equilibria λ_i^*, ϵ_i^* respectively [51]. Next we will show $\epsilon_i^* = \lambda_i^* = 0$ by contradiction.

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 τ^*, λ^*, ϵ^*, which satisfy all the assumptions in theorem 2.3.1. Suppose at least one user r has a nonzero equilibrium ϵ_r^*. From theorem 2.3.1, we know $z_r(t)$ also asymptotically converges to zero and enters the $|z_r(t)| \le \underline{\epsilon}_r$ 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 will 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 2.3.2, 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. ■

The following lemma provides a lower bound on $\dot{L}(x; \tau, \lambda)$ for fixed barrier parameters. This bound will be later used in the proof of theorem 2.3.4.

Lemma 2.3.3 *Under the assumptions of theorem 2.3.1, for all* $t \ge 0$*,*

$$-\frac{5}{2}\sum_{i\in\mathcal{S}} z_i^2(t) \le \frac{dL(x(t); \tau, \lambda)}{dt} \le 0 \tag{2.31}$$

Proof: Remember

$$\begin{aligned}
& -\dot{L}(x;\tau,\lambda) \\
= \; & \sum_{i=1}^{N} z_i[\nabla U_i(x_i) + \frac{\lambda_i}{x_i} - \sum_{j=1}^{M} \mu_j A_{ji}] && (2.32) \\
\leq \; & \frac{1}{2}\sum_{i=1}^{N} \left\{ z_i^2 + [z_i + \sum_{j=1}^{M} \hat{\mu}_j A_{ji} - \sum_{j=1}^{M} \mu_j A_{ji}]^2 \right\} && (2.33) \\
\leq \; & \frac{3}{2}\sum_{i=1}^{N} z_i^2 + \sum_{i=1}^{N} \left\{ [\sum_{j=1}^{M} \hat{\mu}_j A_{ji} - \sum_{j=1}^{M} \mu_j A_{ji}]^2 \right\} && (2.34)
\end{aligned}$$

As a result of the events in theorem 2.3.1, the right-hand side of equation 2.13 in the proof is nonnegative, which is

$$\sum_{i=1}^{N} z_i^2 - \sum_{i=1}^{N} \left\{ [\sum_{j=1}^{M} \hat{\mu}_j A_{ji} - \sum_{j=1}^{M} \mu_j A_{ji}]^2 \right\} \geq 0 \tag{2.35}$$

This means

$$-\dot{L}(x;\tau,\lambda) \leq \frac{3}{2}\sum_{i=1}^{N} z_i^2 + \sum_{i=1}^{N} z_i^2 = \frac{5}{2}\sum_{i=1}^{N} z_i^2 \tag{2.36}$$

which completes the proof. ■

We can now show that algorithms 2.3.1-2.3.2 asymptotically converge to the solution of the NUM problem. 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 barrier parameters. Since there are an infinite number of barrier parameter switches, we can show that the bound in lemma 2.3.3 asymptotically converges to zero, thereby showing that $\dot{L}$ converges to zero and thus establishing the convergence of the algorithm.

Theorem 2.3.4 *Under the assumptions of U_i, A, and ρ in theorem 2.3.1, the data rates $x(t)$ generated by algorithms 2.3.1- 2.3.2 converge asymptotically to the unique solution of the NUM problem.*

Proof: By lemma 2.3.2, there are infinite user switches for each user. Let $\{T[k]\}_{k=0}^{\infty}$ be the nondecreasing sorted sequence of all the elements in $\{T_i^{\lambda}[k]\}_{k=0}^{M_i^S}$ 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 barrier switch occurs for any user. On $[T[k], T[k+1])$, we have fixed parameter τ, λ, ϵ. By lemma 2.3.3, we have

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

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

$$\tilde{\epsilon}_i(t) = \overline{\epsilon}_i[k-1], \quad t \in [T_i^{\lambda}[k], T_i^{\lambda}[k+1]) \tag{2.37}$$

Since $T_i^{\lambda}[k]$ is the time instant when the kth barrier 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, $\dot{L}(x;\tau,\lambda)$ also converges to zero as $t \to \infty$.

Remember equation 2.24 and the user and link events in algorithms 2.3.1-2.3.2, this immediately implies $\forall i \in \mathcal{S}$, $\forall j \in \mathcal{L}$

$$\lim_{t\to\infty}\{z_i^2(t) - \rho\hat{z}_i^2(t)\} = 0 \tag{2.38}$$

$$\lim_{t\to\infty}\{\rho\sum_{i\in\mathcal{S}_j}\frac{1}{\overline{L}}\hat{z}_i^2(t) - \overline{LS}(\mu_j(t) - \hat{\mu}_j(t))^2\} = 0 \tag{2.39}$$

From equation 2.38 and the fact that $\lim_{t\to\infty} z_i(t) = 0$, we have $\lim_{t\to\infty}\hat{z}_i(t) = 0$ for $i \in \mathcal{S}$. When combined with equation 2.39, we obtain $\lim_{t\to\infty}|\mu_j(t) - \hat{\mu}_j(t)| = 0$.

This means

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

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. ■

2.4 Simulation

This section presents simulation results. We compare the number of message exchanges of our event-triggered barrier algorithm against the dual decomposition algorithm and our event-triggered augmented Lagrangian algorithm in [57]. Simulation results show that our event-triggered barrier algorithm reduces the number of message exchanges by up to two order magnitude when compared to dual decomposition, and has comparable performance as the algorithm in [57]. Moreover, our algorithm is scale free with respect to network size. The remainder of this section is organized as follows: Subsection 2.4.1 discusses the simulation setup. Subsection 2.4.2 discusses the effect of the desired precision of the solution on the performance of the algorithm. The effect of different tradeoff parameter ρ is discussed in subsection 2.4.3. Simulation results on broadcast periods of our event-triggered algorithm are shown in subsection 2.4.4. The scalability results with respect to $\overline{S}$ and $\overline{L}$ are presented in subsection 2.4.5 and 2.4.6, respectively. An example where communication delay is taken into consideration is shown in subsection 2.4.7. Finally in subsection 2.4.8 we discuss the effect of different barrier parameters on the performance of the algorithm.

2.4.1 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, all three 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)

$$e(k) = \left|\frac{U(x(k)) - U^*}{U^*}\right| \tag{2.40}$$

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 all 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 the 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 barrier 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. We should point out that since we are comparing a primal algorithm (our event-triggered algorithm) with a dual algorithm, and they run at different time-scales, iteration number is then a more appropriate measure of convergence than time [15] [26].

The default settings for simulation are as follows: For all algorithms, the 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 price $p_j = 0$ for $j \in \mathcal{L}$, and the step size γ is calculated using equation 1.14. In our event-triggered 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. Other parameters include $\rho = 0.5$, $e_d = 1\%$, $M = 60$, $N = 150$, $\overline{L} = 8$, $\overline{S} = 15$.

2.4.2 Effect of desired solution accuracy

This subsection discusses the effect of the desired precision of the solution on the performance of the algorithm.

Since our event-triggered algorithm is based on the barrier method, it is reasonable to expect that the algorithm will not have good performance if we require high precision solution, i.e., e_d extremely small. To see the effect of e_d on the algorithm, we vary e_d from 0.1% to 10%, while keeping all other parameters unchanged. The resulting figure 2.2 plots K as a function of e_d (in logarithm scale) for both dual decomposition and our event-triggered algorithm. The dotted line represents the dual decomposition algorithm, and the circled line corresponds to our event-triggered algorithm.

As we can see, the event-triggered scheme has a much smaller K than its competitor when $e_d \geq 1\%$. As e_d increases from 1% to 10%, K decreases from 3075 to 351 for the dual decomposition algorithm, while for the event-triggered scheme, K decreases from 186 to 25. Over this region, the event-triggered algorithm is about 15 times faster than dual decomposition. However, as expected, when e_d is extremely small, i.e., in this case less than 0.2%, the event-triggered scheme does not show significant advantage over dual decomposition. When $e_d < 0.15\%$, the dual decomposition works even better than event-triggering. This is a result of the underlying barrier methods used.

2.4.3 Effect of tradeoff coefficient ρ

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

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

To see the effect of different ρ on the algorithm, we vary ρ from 0.01 to 0.99, while keeping all other parameters unchanged. The resulting figure 2.3 plots the

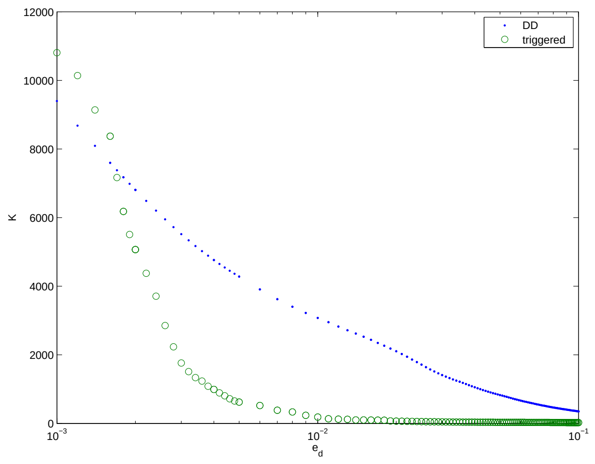

Figure 2.2. Iteration number K as a function of e_d for both algorithms.

triggered events count as a function of ρ. The dotted line at the bottom represents the triggered events count for all users, the middle dotted line represents the triggered events count for all links, while the solid line above corresponds to the total triggered events count.

There are two things we can see in figure 2.3. First, the user events count is much less than the link events count over the entire region $\rho \in (0,1)$. This means in our event-triggered algorithm, the major message exchanges are the links broadcasting their new feedback signal $\hat{\mu}_j$ back to the users. Second, the user events count shows almost linear increase with respect to ρ, while the link events count variation is insensitive to the changes in ρ. Since the link events count contributes to the major part in the total events count, we believe our algorithm is rather insensitive to the choice of ρ, as shown in the figure.

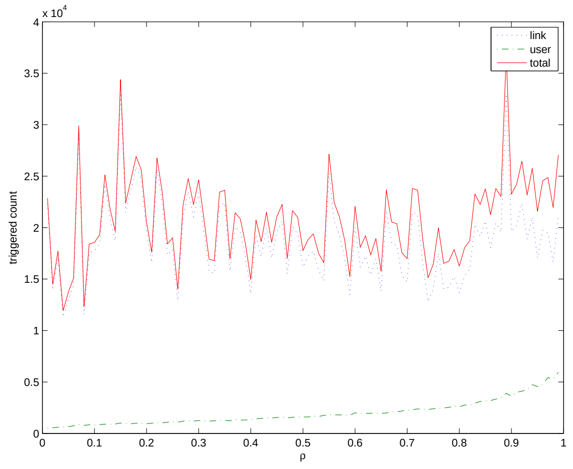

Figure 2.3. Triggered events count as a function of ρ.

2.4.4 Broadcast periods of the event-triggered barrier algorithm

In this subsection we present simulation results on the broadcast periods of our event-triggered algorithm. The simulation ran for 3.12s. For reference, for the same network, the average broadcast period for the dual decomposition is 0.0010, and the average broadcast period for the distributed barrier method without triggering is 0.0017. In our event-triggered algorithm, there are a total of 9273 link events, and 1916 user events. So the links have an average broadcast period of 0.0202, and the users have an average broadcast period of 0.2443. To see how the broadcast periods vary for each user or link, we pick a user and two links in the network, and their broadcast results are shown in the following figures.

Figure 2.4 shows the broadcast periods results of one user. The left plot in figure 2.4 is the time history of broadcast periods generated by its user event. The right plot is the histogram of the broadcast periods. The broadcast periods range between 0.0132 and 0.8900. This user was triggered 16 times, with an average broadcast period of 0.1926.

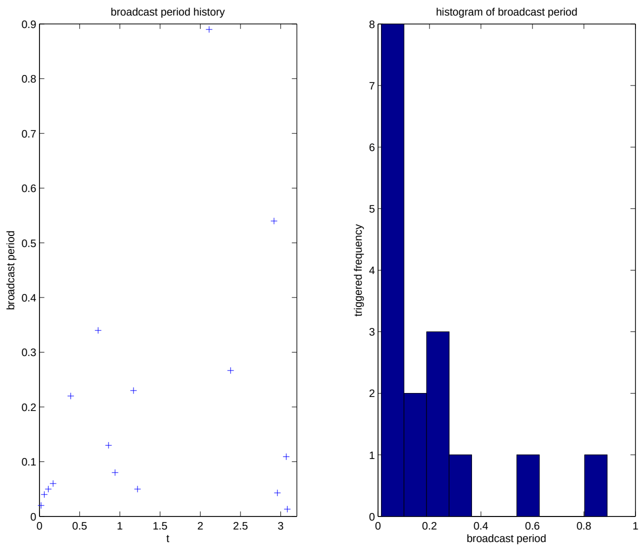

Figure 2.4. Broadcast results for one user

We also generate such figures for an inactive link and an active link. Figure 2.5 shows the broadcast periods results of an inactive link, while figure 2.6 corresponds to an active link.

For the inactive link in figure 2.5, The sampling periods range between 0.0131 and 0.7300. This link was triggered 15 times, with an average broadcast period of 0.1869.

For the active link in figure 2.6, The sampling periods range between 0.0033 and 0.5500. This link was triggered 239 times, with an average broadcast period

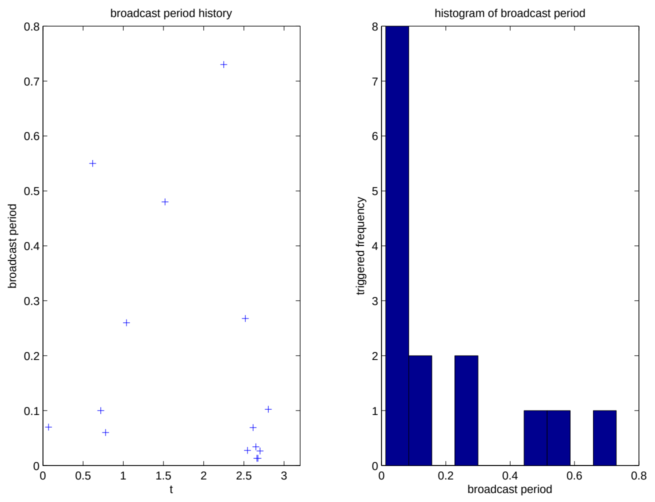

Figure 2.5. Broadcast results for an inactive link

of 0.0130.

As we can see from the above three figures, for the users and inactive links, the number of triggered events are relatively small, and they have long average broadcast periods. For the active links, the number of triggered events are large, which result in short average broadcast periods. The active links broadcasting their link feedback signals to their users contribute to the major part of the number of message exchanges. We can also see that the broadcast periods in figure 2.6 are 'clustered'. Each 'cluster' here usually corresponds to an time interval over which the barrier parameters do not change for this link or its users. When the broadcast periods move from one 'cluster' to another, it usually corresponds to some change in the barrier parameters.

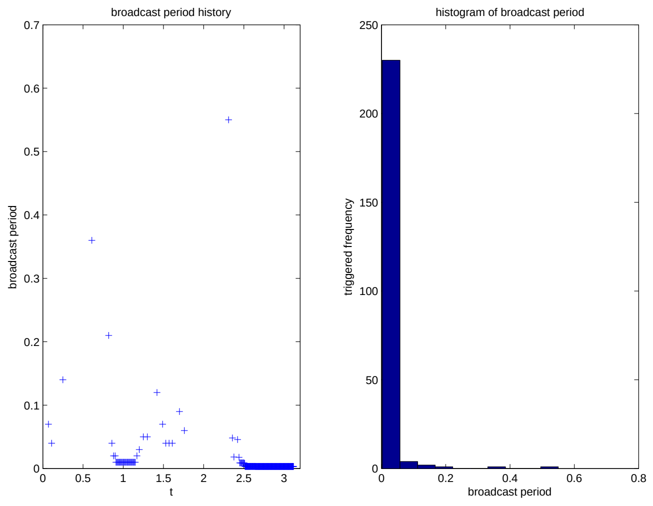

Figure 2.6. Broadcast results for an active link

2.4.5 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 $\overline{S}$, all three algorithms were run 1000 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 algorithms.

Figure 2.7 plots the iteration number K (in logarithm scale) as a function of $\overline{S}$ for all three algorithms. The asterisks above represent m_K for dual decomposition, the circles in the middle correspond to our event-triggered barrier algorithm, while the diamonds below are m_K for the event-triggered augmented Lagrangian method in [57]. 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.

For our event-triggered barrier algorithm, when $\overline{S}$ increases from 7 to 26,

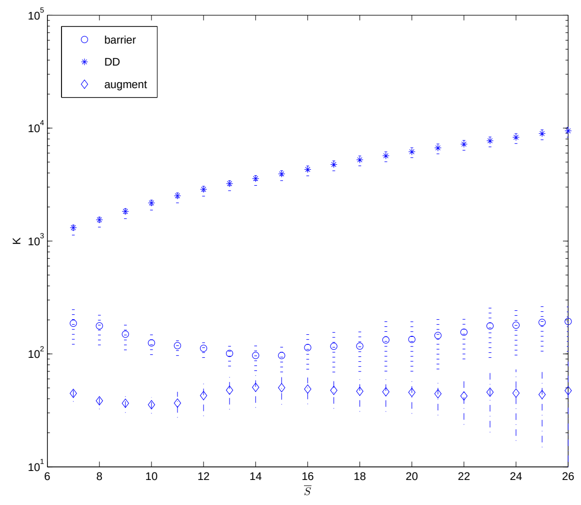

Figure 2.7. Iteration number K as a function of $\overline{S}$ for all algorithms.

similar to the event-triggered augmented Lagrangian algorithm, m_K does not show noticeable increase, and it varies between 90 and 189. The standard deviation σ_K varies between 22 and 78. For dual decomposition, m_K increases from 1.3103×10^3 to 9.5412×10^3. σ_K at the same time increases from 0.187×10^3 to 1.171×10^3. Our event-triggered barrier algorithm is up to 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 virtually scale-free. For comparison, in the event-triggered augmented Lagrangian algorithm, m_K varies between 35 and 50, and σ_K at the same time varies between 6 and 33. This means although the event-triggered barrier algorithm is significantly faster than dual decomposition, it is slower than our later algorithm in [57].

2.4.6 Scalability with respect to $\overline{L}$

This simulation is similar to subsection 2.4.5 except that we vary $\overline{L}$ from 4 to 18 instead of $\overline{S}$.

Figure 2.8 plots K (in logarithm scale) as a function of $\overline{L}$ for all algorithms. For our event-triggered algorithm, when $\overline{L}$ increases from 4 to 18, m_K does not show noticeable increase, and it varies between 93 and 228. σ_K varies between 22 and 86. We notice that, however, our event-triggered algorithm has worse performance when $\overline{L}$ is small. For dual decomposition, m_K increases from 1.8074×10^3 to 8.6312×10^3. σ_K at the same time increases from 0.173×10^3 to 1.143×10^3. Our event-triggered algorithm is up to 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 virtually scale-free. For comparison, in the event-triggered augmented Lagrangian algorithm, m_K varies between 35 and 68, and σ_K at the same time varies between 12 and 34. Similar to the previous subsection, although the event-triggered barrier algorithm is significantly faster than dual decomposition, it is slower than our later algorithm in [57].

The reason that the event-triggered barrier algorithm has worse performance at small $\overline{L}$ is because the parameters we choose work poorly for those cases. The choice of barrier parameters can sometimes greatly affect the performance of our algorithm, as we will show in subsection 2.4.8.

2.4.7 An example considering transmission delay

Transmission delays usually exist in networked systems. In our event-triggered algorithm presented in the previous section, we did not consider transmission

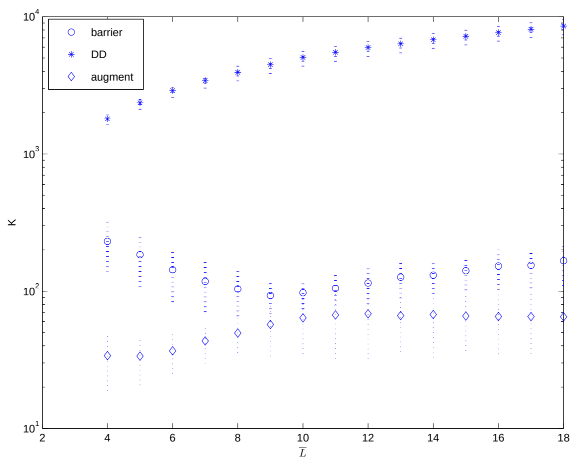

Figure 2.8. Iteration number K as a function of $\overline{L}$ for all algorithms.

delays in the network. Due to the complexity of the algorithm, it is difficult to analyze the effect of delays analytically. However, simulation results show that the algorithm still converges when the delay is not too big.

In this subsection we consider the scenario when each $\hat{z}_i$ or $\hat{\mu}_j$ packet experiences a random transmission delay ranging from 0.002s to 0.02s. The system reaches the e_d neighborhood in 4.3s. Figure 2.9 has two plots. The dotted plot above corresponds to the case when transmission delays are taken into consideration, and the solid plot below corresponds to the case when there is no transmission delay. As we see, the algorithm converges in both cases. However, in the case without delay, there are only 180 iterations, and in the case with delay, there are 310 iterations. The increase in iteration number is expected with the use of outdated information. The simulation clearly demonstrated the effectiveness of our event-triggered algorithm with the presence of transmission delay.

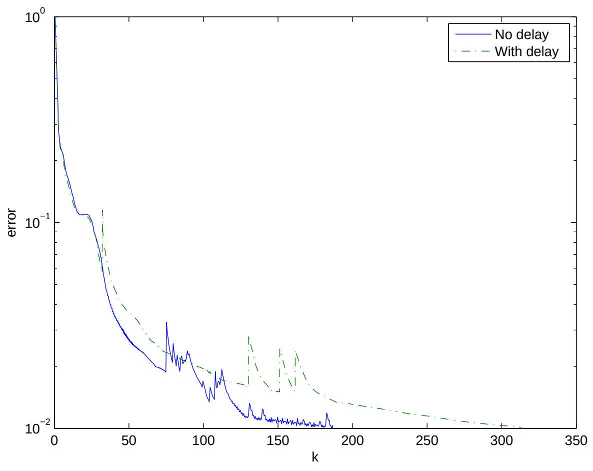

Figure 2.9. Error $e(k)$ as a function of iteration number k

2.4.8 Effect of barrier parameters

In this subsection we consider the effect of different barrier parameters on the performance of our event-triggered algorithm. Remember by default, the sequences $\{\overline{\lambda}_i[k]\}_{k=0}^{\infty}$, $\{\overline{\epsilon}_i[k]\}_{k=0}^{\infty}$, $\{\overline{\tau}_j[k]\}_{k=0}^{\infty}$ are chosen as $\{0.1^k\}_{k=0}^{\infty}$, $\{5\times 0.1^k\}_{k=0}^{\infty}$, $\{0.1^k\}_{k=0}^{\infty}$, respectively. Here we consider two different scenarios from the default setup. In the first scenario, we choose $\{\overline{\tau}_j[k]\}_{k=0}^{\infty}$ as $\{\alpha^k\}_{k=0}^{\infty}$, where $\alpha = \overline{\tau}_j[k+1]/\overline{\tau}_j[k]$ varies from 0.02 to 0.5, while keeping all other parameters unchanged. α measures how fast we lower the barrier level, and let the trajectory of iterates approach the boundary of the feasible set. The smaller α is, the faster we lower the barrier level. In the second scenario, we choose $\{\overline{\epsilon}_i[k]\}_{k=0}^{\infty}$ as $\{\beta\times 0.1^k\}_{k=0}^{\infty}$, where β varies from 0.1 to 30, while keeping all other parameters unchanged. β measures how far away we allow the iterates to stay off-course from the primal central path. The smaller β is, the less off-course the iterates are allowed to be.

Figure 2.10 plots K as a function of α in the first scenario. Over the region $[0.05, 0.25]$, K is relatively small, and varies from 186 to 580. On region $[0.25, 0.5]$,

K increases dramatically, from 440 to 8000 as α increases. On region $(0, 0.05]$, K increases as α decreases.

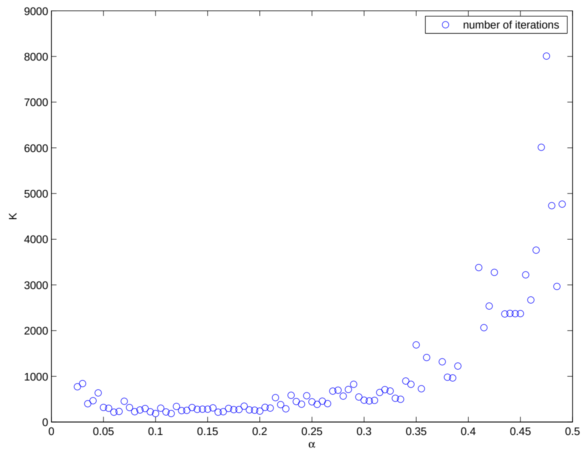

Figure 2.10. Iteration number K as a function of $\alpha = \overline{\tau}_j[k+1]/\overline{\tau}_j[k]$.

Figure 2.11 plots K as a function of β in the second scenario. Over the region $[0.4, 6]$, K is relatively small, and varies from 147 to 470. On region $[6, 30]$, K shows linear increase in the trend. On region $[0.1, 0.4]$, K increases dramatically as β decreases. When $\beta = 0.1$, $K = 1155$.

The phenomenon in both figure 2.10 and 2.11 can be explained by the barrier method itself. In figure 2.10, when α increases, we are slowing down the speed of lowering the barrier level, so the algorithm will in general converge slower, which results in larger K. However, if we choose α too small, then the next desired point on the central path might be very far away from the current iterate, which results

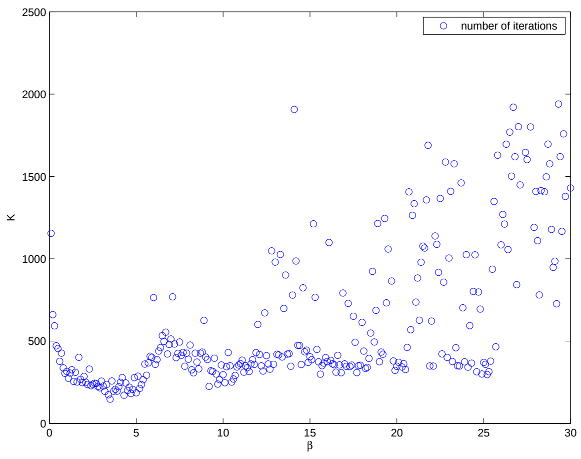

Figure 2.11. Iteration number K as a function of β.

in slow convergence as well. This means α can neither be too large nor too small, which is shown in figure 2.10. In figure 2.11, when β increases, we are allowing the iterates to stay further away from the primal central path, which may result in landing on the boundaries of the feasible set prematurely. In this case, the barrier methods will then progress very slowly. This also explains why in region $[6, 30]$ of figure 2.11, we do not have a large K in all cases. Because in some cases, the iterates may not have premature landing. However, as β increases, the possibility of premature landing increases as well. When β is chosen too small, the iterates are then forced to stay very close to the primal central path, which may result in large number of iterations for each set of barrier parameters. This will slow down the convergence of the algorithm as well. So just like α, β can not be too large or too small either.

2.5 Conclusion

This chapter presents an event-triggered distributed optimization algorithm based on the barrier methods. We use the NUM problem as an example to illustrate the idea of event-triggering in distributed optimization. The chapter 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 up to two orders of magnitude when compared to existing dual decomposition algorithms.

This is our very first approach to solve the distributed optimization problems using the event-triggered idea, and it shows significantly reduction in the communication cost of the network. However, as we show in the chapter, the algorithm suffers from issues like the need for an initial feasible point, and performance sensitive to parameter choices. All these issues limit the usefulness the algorithm. In our later work (which will be discussed in chapter 3), we propose an event-triggered algorithm based on the augmented Lagrangian method, which does not suffer from the issues mentioned above, while exhibiting even better performance.

CHAPTER 3

Event-triggered distributed optimization using augmented Lagrangian methods

3.1 Introduction

This chapter uses the NUM problem formulation as an example and presents two distributed algorithms based on the augmented Lagrangian methods that use event-triggered message passing and proves their convergence. One of the algorithm is a primal algorithm, and the other is a primal-dual algorithm. For the primal-dual algorithm, we consider scenarios when the network has data dropouts, and give an upper bound on the largest number of successive data dropouts each link can have, while ensuring the asymptotic convergence of the algorithm. A state-dependent lower bound on the broadcast period and an upper bound on the transmission delay the network can tolerate while ensuring convergence are also given. Simulation results show that both algorithms have a message passing complexity that is up to two orders of magnitude lower than dual decomposition algorithms, and are scale-free with respect to two measures of network size $\overline{L}$ and $\overline{S}$. The primal algorithm in this chapter is similar to the algorithm in chapter 2. However, in chapter 2, 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 , need for an initial feasible point and sensitive to the choice of parameters. The event-triggered algorithms presented in this chapter do not have these issues.

The rest of the chapter is organized as follows. Two event-triggered algorithms, the primal algorithm, and the primal-dual algorithm, will be presented in section 3.2 and section 3.3, respectively. Simulation results are shown in section 3.4, and section 3.5 concludes the chapter.

3.2 The Primal algorithm

This section presents the event-triggered distributed primal algorithm. The algorithm is based on the augmented Lagrangian method for the NUM problem, which will be first discussed in subsection 3.2.1. Subsection 3.2.2 then presents the event-triggered distributed primal algorithm, and proves its convergence.

3.2.1 Basic Primal Algorithm

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 $c_j - a_j^T x \geq 0$, $\forall j \in \mathcal{L}$ by

$$a_j^T x - c_j + s_j = 0, \quad s_j \geq 0, \quad \forall j \in \mathcal{L} \tag{3.1}$$

The *augmented cost* is then

$$L(x, s; \lambda, w) = -\sum_{i \in \mathcal{S}} U_i(x_i) + \sum_{j \in \mathcal{L}} \lambda_j (a_j^T x - c_j + s_j) + \frac{1}{2} \sum_{j \in \mathcal{L}} \frac{1}{w_j} (a_j^T x - c_j + s_j)^2 \tag{3.2}$$

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 $c_j - a_j^T x \geq 0$ in the Karush-Kuhn-Tucker conditions of the NUM problem. λ_j is an estimate of λ_j^* and $\lambda = [\lambda_1, \cdots, \lambda_M]$. The vector $a_j^T = [A_{j1}, \cdots, A_{jN}]$ is the jth row of the routing matrix A.

$L(x, s; \lambda, w)$ is a continuous function of x and s for fixed λ and w. It is shown [8] that

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

where we define *the augmented Lagrangian function* associated with the NUM problem as

$$L_p(x; \lambda, w) = -\sum_{i\in\mathcal{S}} U_i(x_i) + \sum_{j\in\mathcal{L}} \psi_j(x; \lambda, w) \tag{3.3}$$

Here we have

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

The primal algorithm based on the augmented Lagrangian method solves the NUM problem by approximately minimizing $L_p(x; \lambda[k], \overline{w}[k])$ for sequences of $\{\overline{w}[k]\}_{k=0}^{\infty}$ and $\{\lambda[k]\}_{k=0}^{\infty}$. Let $x^*[k]$ denote the approximate minimizer for $L_p(x; \lambda[k], \overline{w}[k])$. The method in [8, 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 $\lambda_j[k+1] = \max\{0, \lambda_j[k] + \frac{1}{\overline{w}_j[k]}(a_j^T x^*[k] - c_j)\}$.

In our work in [57], we gave a primal algorithm based on the augmented Lagrangian method for the NUM problem that converges to the exact minimizer of the NUM problem. However, in many scenarios, it usually suffices to obtain an approximate minimizer to the problem. So instead of minimizing $L_p(x; \lambda[k], \overline{w}[k])$ for sequences of $\{\overline{w}[k]\}_{k=0}^{\infty}$ and $\{\lambda[k]\}_{k=0}^{\infty}$, we are only considering the problem of minimizing $L_p(x; \lambda, w)$ for fixed λ and w in the discussion here. If $\lambda_j = 0$ and w_j is sufficiently small, the minimizer of $L_p(x; \lambda, w)$ will be a good approximation to the solution of the original NUM problem. We can choose w_j to be small enough such that the approximation lies within the desirable neighborhood of the solution of NUM.

The **basic primal algorithm** for the NUM problem is given as follows:

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

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

$$\begin{aligned} x &= \max\left\{0, x^0 - \gamma \nabla_x L_p(x^0; \lambda, w)\right\} \qquad (3.4) \\ x^0 &= x \end{aligned}$$

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

The computations above can be easily distributed among the users and links. We will see how they are distributed in our event-triggered distributed implementation of the algorithm in section 3.2.2.

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 subsection presents an event-triggered distributed implementation of the basic primal algorithm presented in this subsection.

3.2.2 Event-triggered Distributed Primal Algorithm

Implementing the primal algorithm in section 3.2.1 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 subsection determines such an event threshold condition and gives an event-triggered distributed primal algorithm for solving problem 1.1.

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

$$\begin{aligned} x_i(t) &= -\int_0^t \left(\nabla_{x_i} L_p(x(\tau);\lambda,w)\right)^+_{x_i(\tau)} d\tau \\ &= \int_0^t \left(\frac{\partial U_i(x_i(\tau))}{\partial x_i} - \sum_{j\in\mathcal{L}_i} \mu_j(\tau)\right)^+_{x_i(\tau)} d\tau \end{aligned} \tag{3.5}$$

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

$$\mu_j(t) = \max\left\{0, \lambda_j + \frac{1}{w_j}\left(\sum_{i\in\mathcal{S}_j} x_i(t) - c_j\right)\right\} \tag{3.6}$$

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

$$(f(x))^+_x = \begin{cases} 0, & \text{if} \quad x = 0 \quad \text{and} \quad f(x) < 0 \\ f(x), & \text{otherwise} \end{cases} \tag{3.7}$$

The positive projection used in equation 3.5 guarantees the user rate $x_i(t)$ is always nonnegative along the trajectory.

Equation 3.5 is the continuous-time version of the update in equation 3.4. Note that in equation 3.5, 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 3.6, 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 3.5. 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

$$\hat{\mu}_j(t) = \mu_j(T_j^L[\ell]) \tag{3.8}$$

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 3.5 takes the form

$$x_i(t) \;=\; \int_0^t \left(\frac{\partial U_i(x_i(\tau))}{\partial x_i} - \sum_{j \in \mathcal{L}_i} \hat{\mu}_j(\tau) \right)^+_{x_i(\tau)} d\tau \tag{3.9}$$

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. In the event-triggered primal algorithm, 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

$$z_i(t) \;=\; \dot{x}_i(t) = \left(\nabla U_i(x_i(t)) - \sum_{j \in \mathcal{L}_i} \hat{\mu}_j(t) \right)^+_{x_i(t)} \tag{3.10}$$

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

$$\hat{z}_i(t) = z_i(T_i^S[\ell]) \tag{3.11}$$

for all $\ell = 0, \cdots, \infty$ and any $t \in [T_i^S[\ell], T_i^S[\ell+1])$. In the proposed event-triggering primal algorithm, links will use the sampled user state, $\hat{z}_i$, to help determine when they should transmit their states back to the user. Here we assume that there is no transmission delay in each $\hat{\mu}_j(t)$ or $\hat{z}_i(t)$ broadcast.

Next we will state the main theorem of this subsection.

Theorem 3.2.1 *Consider the Lagrangian in equation 3.3 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 3.9 with sampled link states given by equation 3.8. For each $i \in \mathcal{S}$ let the user state $z_i(t)$ satisfy equation 3.10 and assume link j's measurement of the user state satisfies equation 3.11.*

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

$$z_i^2(t) - \rho \hat{z}_i^2(t) \geq 0 \tag{3.12}$$

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

$$\rho \sum_{i \in \mathcal{S}_j} \frac{1}{\overline{L}} \hat{z}_i^2(t) - \overline{LS}\,(\mu_j(t) - \hat{\mu}_j(t))^2 \geq 0 \tag{3.13}$$

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* $L_p(x; \lambda, w)$. ■

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

$$\begin{aligned} -\dot{L}_p(x; \lambda, w) &= -\sum_{i=1}^{N} \frac{\partial L_p}{\partial x_i} \frac{dx_i}{dt} \\ &= \sum_{i=1}^{N} z_i [\nabla U_i(x_i) - \sum_{j=1}^{M} \mu_j A_{ji}] & (3.14) \\ &\geq \sum_{i=1}^{N} \left\{ \frac{1}{2} z_i^2 - \frac{1}{2} [\sum_{j=1}^{M} \mu_j A_{ji} - \sum_{j=1}^{M} \hat{\mu}_j A_{ji}]^2 \right\} & (3.15) \\ &\geq \sum_{i=1}^{N} \left\{ \frac{1}{2} z_i^2 - \frac{1}{2} [\sum_{j=1}^{M} (\mu_j - \hat{\mu}_j) A_{ji}]^2 \right\} & (3.16) \end{aligned}$$

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

$$-\left[\sum_{j=1}^{M} (\mu_j - \hat{\mu}_j) A_{ji} \right]^2 \geq -|\mathcal{L}_i| \sum_{j=1}^{M} [(\mu_j - \hat{\mu}_j) A_{ji}]^2 \tag{3.17}$$

we have

$$
\begin{aligned}
-\dot{L}_p(x;\lambda,w) &\geq \frac{1}{2}\sum_{i=1}^{N} z_i^2 - \frac{1}{2}\sum_{i=1}^{N}\left\{|\mathcal{L}_i|\sum_{j=1}^{M}[(\mu_j-\hat{\mu}_j)A_{ji}]^2\right\} \\
&= \frac{1}{2}\sum_{i=1}^{N} z_i^2 - \frac{1}{2}\sum_{j=1}^{M}\left\{(\mu_j-\hat{\mu}_j)^2\sum_{i=1}^{N}|\mathcal{L}_i|A_{ji}^2\right\} \qquad (3.18) \\
&\geq \frac{1}{2}\sum_{i=1}^{N} z_i^2 - \frac{1}{2}\sum_{j=1}^{M}\overline{LS}(\mu_j-\hat{\mu}_j)^2 \qquad (3.19)
\end{aligned}
$$

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

$$
\frac{1}{2}\rho\sum_{i=1}^{N}\hat{z}_i^2 = \frac{1}{2}\rho\sum_{i=1}^{N}\overline{L}\frac{1}{\overline{L}}\hat{z}_i^2 = \frac{1}{2}\rho\sum_{j=1}^{M}\sum_{i=1}^{N}\frac{1}{\overline{L}}\hat{z}_i^2A_{ji} + \frac{1}{2}\rho\sum_{i=1}^{N}(\overline{L}-|\mathcal{L}_i|)\frac{1}{\overline{L}}\hat{z}_i^2 \qquad (3.20)
$$

The last equality holds since

$$
\sum_{i=1}^{N}|\mathcal{L}(i)|\frac{1}{\overline{L}}\hat{z}_i^2 = \sum_{i=1}^{N}\sum_{j=1}^{M}\frac{1}{\overline{L}}\hat{z}_i^2A_{ji} = \sum_{j=1}^{M}\sum_{i=1}^{N}\frac{1}{\overline{L}}\hat{z}_i^2A_{ji} \qquad (3.21)
$$

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

$$
\begin{aligned}
&-\dot{L}_p(x;\lambda,w) \\
\geq\ & \frac{1}{2}\sum_{i=1}^{N} z_i^2 - \frac{1}{2}\rho\sum_{i=1}^{N}\hat{z}_i^2 + \frac{1}{2}\rho\sum_{i=1}^{N}\hat{z}_i^2 - \frac{1}{2}\sum_{j=1}^{M}\overline{LS}(\mu_j-\hat{\mu}_j)^2 \qquad (3.22) \\
\geq\ & \frac{1}{2}\sum_{i=1}^{N}[z_i^2-\rho\hat{z}_i^2] + \frac{1}{2}\sum_{j=1}^{M}\left\{\rho\sum_{i\in\mathcal{S}_j}\frac{1}{\overline{L}}\hat{z}_i^2 - \overline{LS}(\mu_j-\hat{\mu}_j)^2\right\} \qquad (3.23)
\end{aligned}
$$

which immediately suggests that 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 3.12 and 3.13 for all $\ell = 0, 1, 2, ..., \infty$, and any $i \in \mathcal{S}$, $j \in \mathcal{L}$, then $\dot{L}_p(x;\lambda,w) \leq 0$ 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, $L_p(x;\lambda,w)$ is strictly convex in x. It thus has a unique minimizer. Suppose $x^*(\lambda,w)$ is this minimizer, and the corresponding Lagrangian is $L_p(x^*;\lambda,w)$. Define $V(x) = L_p(x;\lambda,w) - L_p(x^*;\lambda,w)$. It is trivial to see $V(x)$ is a Lyapunov function for the system. Moreover, $\dot{V}(x) = 0$ means $\dot{L}_p(x;\lambda,w) = 0$. The only scenario this can happen is

$$z_i = \hat{z}_i = 0, \quad \forall i \in \mathcal{S}, \quad \mu_j = \hat{\mu}_j, \quad \forall j \in \mathcal{L} \tag{3.24}$$

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

Theorem 3.2.1 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 3.12 and 3.13 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 3.12 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 links $j \in \mathcal{L}_i$. 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 users $i \in \mathcal{S}_j$.

In theorem 3.2.1, ρ models the tradeoff between triggering the user events and triggering the link events. When ρ is large, we would expect more user events and less link events. On the contrary, when ρ is small, we would expect more link events and less user events.

The threshold conditions shown in equations 3.12-3.13 provide the basis for an event-triggered implementation of the basic primal algorithm presented earlier in subsection 3.2.1. Next we will present such an event-triggered distributed primal algorithm.

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

Each 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.

Algorithm 3.2.1 User i's Update Algorithm

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

2. **State Initialization:** *Upon receiving link state $\mu_j(T)$ from $j \in \mathcal{L}_i$, set $\hat{\mu}_j = \mu_j(T)$. Initialize user state to*

$$z_i(T) = \left(\nabla U_i(x_i^0) - \sum_{j \in \mathcal{L}_i} \hat{\mu}_j \right)^+_{x_i(T)} \qquad (3.25)$$

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*

$$x_i(t) = \int_T^t z_i(\tau)d\tau \tag{3.26}$$

$$z_i(t) = \left(\nabla U_i(x_i(t)) - \sum_{j\in\mathcal{L}_i}\hat{\mu}_j\right)^+_{x_i(t)} \tag{3.27}$$

$$x_i(T) = x_i^0 \tag{3.28}$$

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 $\hat{\mu}_j = \mu_j^+(T^+)$.

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 3.2.2 Link j's Update Algorithm

1. **Parameter Initialization:** *Set $T = 0$, $w_j > 0$.*

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

$$\mu_j(T) = \max\left\{0, \frac{1}{w_j}\left(\sum_{i\in\mathcal{S}_j} x_i(T) - c_j\right)\right\} \tag{3.29}$$

Transmit $\mu_j(T)$ to all users $i \in \mathcal{S}_j$ and set $\hat{\mu}_j = \mu_j(T)$. Upon receiving user state $z_i(T)$ from $i \in \mathcal{S}_j$, 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 $\hat{z}_i = z_i^+(T^+)$.

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

By theorem 3.2.1, the data rates $x(t)$ generated by algorithms 3.2.1-3.2.2 converge asymptotically to the unique minimizer of $L_p(x; \lambda, w)$, which is an approximate solution to the NUM problem.

In our work in [57], we gave an event-triggered distributed primal algorithm that 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 sequence of approximate minimizers asymptotically approach the solution of the NUM problem. However, in this chapter, we only consider fixed penalty scenarios.

3.3 The Primal-dual algorithm

In the algorithm in chapter 2 and the primal algorithm in section 3.2.2, there is an event associated with each user and link. The interactions between the user events and link events complicate the event-triggered algorithm significantly, and make the analysis of the algorithm very difficult. This section presents a different event-triggered algorithm, namely, the primal-dual algorithm. In the primal-dual algorithm, there are only link events. The algorithm has comparable performance as the primal algorithm, but the simplicity in the event structure enables us to obtain some additional analytical results. Moreover, the primal-dual algorithm in this section is much easier to implement.

The event-triggered distributed primal-dual algorithm is based on the basic primal-dual algorithm for the NUM problem, which will be first discussed in subsection 3.3.1. Subsection 3.3.2 then presents the event-triggered distributed primal-dual algorithm, and proves its convergence. Subsection 3.3.3 considers scenarios when the network has data dropouts, 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 subsection 3.3.2. Subsection 3.3.4 analyzes the broadcast period of the event-triggered primal-dual algorithm, and lower bounds it as a function of the local link state in the network. Finally subsection 3.3.5 studies the effect of transmission delay on the algorithm, and gives an upper bound on the maximum tolerable delay while ensuring the convergence of the event-triggered algorithm.

3.3.1 Basic Primal-dual Algorithm

The event-triggered distributed primal-dual algorithm in this section is also based on the augmented Lagrangian method for the NUM problem. However, it uses the augmented Lagrangian method in a slightly different way than in section 3.2.

Recall that the augmented cost for the NUM problem in equation 3.2 is

$$L(x, s; \lambda, w) = -\sum_{i \in \mathcal{S}} U_i(x_i) + \sum_{j \in \mathcal{L}} \lambda_j (a_j^T x - c_j + s_j) + \frac{1}{2} \sum_{j \in \mathcal{L}} \frac{1}{w_j} (a_j^T x - c_j + s_j)^2 \tag{3.30}$$

In the primal algorithm in section 3.2, we eliminate the dual variable s, and rewrite the augmented cost in equation 3.2 as equation 3.3, which is only a function of the primal variable x for fixed λ and w. That approach gives us the resulting basic primal algorithm in section 3.2. In this section, we are adopting a slightly different approach. To be specific, we deal with the augmented cost in equation 3.30 directly. $L(x, s; \lambda, w)$ is a function of both the primal variable x and dual variable s. We deal with them simultaneously and similarly obtain a basic primal-dual algorithm.

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

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

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

$$\begin{aligned} x &= \max\left\{0, x^0 - \gamma \nabla_x L(x^0, s^0; \lambda, w)\right\} && (3.31) \\ s &= \max\left\{0, s^0 - \gamma \nabla_s L(x^0, s^0; \lambda, w)\right\} && (3.32) \\ (x^0, s^0) &= (x, s) \end{aligned}$$

Again the computations above in equations 3.31-3.32 can be easily distributed among the users and links. We will see how they are distributed in our event-triggered distributed implementation of the algorithm next.

3.3.2 Event-triggered Distributed Primal-dual Algorithm

This subsection presents an event-triggered distributed implementation of the basic primal-dual algorithm in subsection 3.3.1. The idea is similar to subsection 3.2.2, where we apply event-triggering to an underlying gradient following algorithm.

For each link $j \in \mathcal{L}$, we have

$$\begin{aligned} s_j(t) &= -\int_0^t \left(\nabla_{s_j} L(x(\tau), s(\tau); \lambda, w)\right)^+_{s_j(\tau)} d\tau \\ &= \int_0^t (-\mu_j(\tau))^+_{s_j(\tau)} d\tau \end{aligned} \tag{3.33}$$

where

$$\mu_j(t) = \lambda_j + \frac{1}{w_j}\left(\sum_{i \in \mathcal{S}_j} x_i(t) - c_j + s_j(t)\right) \tag{3.34}$$

Unlike in the primal algorithm in subsection 3.2.2, here link $j \in \mathcal{L}$ is associated with a dynamical system which is characterized by equations 3.33-3.34. 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, we still call μ_j as the jth link's *local state*, which serves as the feedback signal to the users in $i \in \mathcal{S}_j$.

Similar to subsection 3.2.2, 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 3.34 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 satisfies

$$\hat{\mu}_j(t) = \mu_j(T_j^L[\ell]) \tag{3.35}$$

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

For each user $i \in \mathcal{S}$ we have

$$x_i(t) \quad = \quad \int_0^t \left(\frac{\partial U_i(x_i(\tau))}{\partial x_i} - \sum_{j \in \mathcal{L}_i} \hat{\mu}_j(\tau) \right)^+_{x_i(\tau)} d\tau \tag{3.36}$$

for all ℓ and any $t \in [T_j^L[\ell], T_j^L[\ell+1])$. Here we assume that there is no transmission delay in each $\hat{\mu}_j(t)$ broadcast.

Similar to the primal algorithm, the user rate update and the link state computation above can again be done in a distributed manner.

In the primal-dual algorithm in this section, links do not need knowledge of the time derivative of users' flow rate, and only link events are needed in the algorithm. Next we will state the main theorem of this subsection.

Theorem 3.3.1 *Consider the Lagrangian in equation 3.30 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 3.33-3.34. For each user $i \in \mathcal{S}$, let the user rate, $x_i(t)$, satisfy equation 3.36 with sampled link states defined in equation 3.35.*

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

$$\rho_j \left[(-\mu_j(t))^+_{s_j(t)} \right]^2 - \frac{1}{2} \overline{LS} \left[\mu_j(t) - \hat{\mu}_j(t) \right]^2 \geq 0 \tag{3.37}$$

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: Define $z_i(t)$ in the same way as in equation 3.10, where $\hat{\mu}_j(t)$ is given in equation 3.35. For all $t \geq 0$ we have

$$\begin{aligned}
& -\dot{L}(x, s; \lambda, w) \\
= \;& -\sum_{i=1}^{N} \frac{\partial L}{\partial x_i}\frac{dx_i}{dt} - \sum_{j=1}^{M} \frac{\partial L}{\partial s_j}\frac{ds_j}{dt} \\
= \;& \sum_{i=1}^{N} z_i \left[\nabla U_i(x_i) - \sum_{j=1}^{M} \mu_j A_{ji} \right] + \sum_{j=1}^{M} (-\mu_j)^+_{s_j}(-\mu_j) \\
\geq \;& \sum_{i=1}^{N} \left\{ \frac{1}{2} z_i^2 - \frac{1}{2} [\sum_{j=1}^{M} (\mu_j - \hat{\mu}_j) A_{ji}]^2 \right\} + \sum_{j=1}^{M} \left[(-\mu_j)^+_{s_j} \right]^2 \\
\geq \;& \frac{1}{2}\sum_{i=1}^{N} z_i^2 - \frac{1}{2}\sum_{j=1}^{M} \overline{LS}(\mu_j - \hat{\mu}_j)^2 + \sum_{j=1}^{M} \left[(-\mu_j)^+_{s_j} \right]^2 && (3.38) \\
= \;& \frac{1}{2}\sum_{i=1}^{N} z_i^2 + \sum_{j=1}^{M} (1 - \rho_j) \left[(-\mu_j)^+_{s_j} \right]^2 \\
& + \sum_{j=1}^{M} \left\{ \rho_j \left[(-\mu_j)^+_{s_j} \right]^2 - \frac{1}{2}\overline{LS}(\mu_j - \hat{\mu}_j)^2 \right\} && (3.39)
\end{aligned}$$

The above inequalities are easy to check using the proof of theorem 3.2.1 and the definition of positive projection in equation 3.7. This immediately suggests us that if the sequences of sampling instants $\{T_j^L[\ell]\}_{\ell=0}^{\infty}$ satisfy the inequality in equation 3.37 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. Using a similar argument as the proof of theorem 3.2.1, we immediately know that $x(t)$ and $s(t)$ asymptotically converge to the unique minimizer of $L(x, s; \lambda, w)$. ■

The threshold condition in equation 3.37 provides the basis for an event-triggered distributed implementation of the basic primal-dual algorithm presented earlier in subsection 3.3.1.

Remember that in the primal algorithm in section 3.2, ρ is a constant for the entire network that tradeoffs between triggering the user events and triggering the link events. Here in theorem 3.3.1, we have a similar parameter ρ_j. 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 3.37. For stability consideration in theorem 3.3.1, it is correct that we can simply choose $\rho_j = 1$. However, as we will see in the next subsection, 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 3.37, 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 3.33-3.34 reduces to a memoryless function. In those situations, if we still use the inequality in equation 3.37 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. Fortunately, this turns out to be not a serious problem here since we are more interested in how fast the primal-dual algorithm enters some neighborhood of the optimal point. This neighborhood can be chosen large enough so that the positive projections will not become active before entering the neighborhood. In all our simulations, this neighborhood is chosen to be within 3% relative error (refer to equation 3.77) around the optimal point, which is rather small. The positive projections are not active before entering this 3% neighborhood in all of our simulations. If we insist on obtaining a more accurate solution, what we can do is, once the projection stays active, then we no longer use the primal-dual algorithm. Instead, we switch to the primal algorithm 3.2.1-3.2.2 in subsection 3.2.2. This will enable us to get a more accurate solution.

In the remaining part of the chapter, we will assume that the positive projection in equation 3.37 cannot be active unless at the minimizer of $L(x, s; \lambda, w)$. In general this assumption is only true with certain choice of penalty coefficient w_j for each link j. However, it is reasonable if we are only interested in converging to some neighborhood of the optimal point, because our analysis only focuses on the behavior of the system before entering this neighborhood. This assumption 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 3.37.

Corollary 3.3.2 *Consider the Lagrangian in equation 3.30 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 3.33-3.34. For each user $i \in \mathcal{S}$, let the user rate, $x_i(t)$, satisfy equation 3.36 with sampled link states defined in equation 3.35.*

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

$$|\mu_j(t) - \hat{\mu}_j(t)| \leq \delta_j |\hat{\mu}_j(t)| \tag{3.40}$$

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

$$\delta_j = \frac{\sqrt{\rho}_j}{\sqrt{\frac{1}{2}\overline{LS}} + \sqrt{\rho}_j} \tag{3.41}$$

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 3.41, equation 3.40 is equivalent to

$$\sqrt{\frac{1}{2}\overline{LS}}|\mu_j(t) - \hat{\mu}_j(t)| + \sqrt{\rho_j}|\mu_j(t) - \hat{\mu}_j(t)| \leq \sqrt{\rho_j}|\hat{\mu}_j(t)| \tag{3.42}$$

Therefore, we have

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

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 3.37 cannot be active unless at the minimizer of $L(x, s; \lambda, w)$, all assumptions of theorem 3.3.1 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 3.37 or 3.40 can both be used as the basis for the event-triggered algorithm. Equation 3.40 is a slightly more conservative condition, and we will use it in our **event-triggered distributed primal-dual algorithm** in the following.

Each user $i \in \mathcal{S}$ executes the following algorithm.

Algorithm 3.3.1 User i's Update Algorithm

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

2. **State Initialization:** *Upon receiving link state $\mu_j(T)$ from $j \in \mathcal{L}_i$, 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 3.3.2 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*

$$\delta_j = \frac{\sqrt{\rho_j}}{\sqrt{\frac{1}{2}\overline{LS}} + \sqrt{\rho_j}} \tag{3.43}$$

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

$$\mu_j(T) = \frac{1}{w_j}\left(\sum_{i \in \mathcal{S}_j} x_i(T) - c_j + s_j^0\right) \tag{3.44}$$

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*

$$s_j(t) = \int_T^t (-\mu_j(\tau))^+_{s_j(\tau)}\, d\tau \tag{3.45}$$

$$\mu_j(t) = \frac{1}{w_j}\left(\sum_{i \in \mathcal{S}_j} x_i(t) - c_j + s_j(t)\right) \tag{3.46}$$

$$s_j(T) = s_j^0 \tag{3.47}$$

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 3.3.2, the data rates $x(t)$ generated by algorithms 3.3.1-3.3.2 converge asymptotically to the unique minimizer of $L(x, s; \lambda, w)$, which is an approximate solution to the NUM problem.

Similar to the primal algorithm 3.2.1-3.2.2, our primal-dual algorithm is also fully distributed. Moreover, there are only link events in the algorithm, which eliminates the need for user event transmissions. This results in a much simpler event-triggered algorithm.

3.3.3 Event-triggering with data dropouts

The discussion in the previous subsection 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 subsection, 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 subsection 3.3.2. 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 3.40, $\hat{\mu}_j(t)$ 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 that the event in equation 3.40 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 3.36 and user's knowledge of the sampled link state in equation 3.35 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 3.3.3 *Consider the Lagrangian in equation 3.30 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 3.33-3.34. For each user $i \in \mathcal{S}$, let the user rate, $x_i(t)$, satisfy equation 3.36 with sampled link states defined in equation 3.35.*

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

$$|\mu_j(t) - \mu_j(r_j[k])| \leq \delta_j |\mu_j(r_j[k])| \tag{3.48}$$

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

$$\delta_j = \frac{\sqrt{\rho}_j}{\sqrt{\frac{1}{2}\overline{LS}} + \sqrt{\rho}_j} \tag{3.49}$$

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

$$d_j \leq D_j(\rho_j) = \log_{(\frac{1}{1-\delta_j})}(1 + \sqrt{\frac{2}{\overline{LS}}}) - 1 \tag{3.50}$$

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

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

Applying equation 3.48 on the previous equation yields

$$|e_j(t)| \leq \delta_j \sum_{p=0}^{k} |\mu_j(r_j[p])| \tag{3.51}$$

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

From equation 3.48, we can easily obtain

$$|\mu_j(r_j[k])| \le \frac{1}{1-\delta_j}|\mu_j(t)| \tag{3.52}$$

for all $t \in [r_j[k], r_j[k+1])$. Applying equation 3.48 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

$$|\mu_j(r_j[p])| \le \left(\frac{1}{1-\delta_j}\right)^{k+1-p}|\mu_j(t)| \tag{3.53}$$

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

Applying equation 3.53 on equation 3.51 yields

$$|e_j(t)| \le \delta_j \sum_{p=0}^{k}\left(\frac{1}{1-\delta_j}\right)^{k+1-p}|\mu_j(t)| = \left[\left(\frac{1}{1-\delta_j}\right)^{k+1} - 1\right]|\mu_j(t)| \tag{3.54}$$

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

$$|e_j(t)| \le \left[\left(\frac{1}{1-\delta_j}\right)^{d_j+1} - 1\right]|\mu_j(t)| \tag{3.55}$$

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

$$|\mu_j(t) - \hat{\mu}_j(t)| \le \left[\left(\frac{1}{1-\delta_j}\right)^{d_j+1} - 1\right]|\mu_j(t)| \le \sqrt{\frac{2}{\overline{LS}}}|\mu_j(t)| \tag{3.56}$$

The last inequality holds by applying equation 3.50.

Remember that we assumed the positive projection on $\mu_j(t)$ is not active unless at the equilibrium, then from the proof of theorem 3.3.1 and apply equation 3.56, we know that for all $t \geq 0$ we have

$$-\dot{L}(x, s; \lambda, w) \geq \frac{1}{2}\sum_{i=1}^{N} z_i^2 - \frac{1}{2}\sum_{j=1}^{M} \overline{LS}(\mu_j - \hat{\mu}_j)^2 + \sum_{j=1}^{M} \mu_j^2 \geq \frac{1}{2}\sum_{i=1}^{N} z_i^2 \tag{3.57}$$

The convergence then follows easily. ■

d_j is link j's largest number of successive data dropouts, and $D_j(\rho_j)$ represents the maximum allowable number of successive data dropouts for link j. This theorem guarantees that algorithm 3.3.1-3.3.2 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 $D_j(\rho_j)$ is a monotone decreasing function in ρ_j, and 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 $\rho_j \to 1$, $D_j(\rho_j) \to 0$.

3.3.4 Broadcast period analysis for the primal-dual algorithm

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

Since in corollary 3.3.2 the algorithm asymptotically converges to the equilibrium, we can thus conclude that the state trajectory of the system always lies within some compact set. Therefore, there always exists a constant $p_i > 0$ such that

$$z_i(t) = \left(\frac{\partial U_i(x_i(t))}{\partial x_i} - \sum_{j \in \mathcal{L}_i} \hat{\mu}_j(t) \right)^+_{x_i(t)} \leq p_i \tag{3.58}$$

for any $t \geq 0$ and $i \in \mathcal{S}$.

We can then derive a lower bound on the broadcast period of each link. Note that we assume there is no data dropouts in the following discussion.

Corollary 3.3.4 *Suppose all the assumptions in corollary 3.3.2 hold, and there is no data dropouts in the system. Also $\forall t \geq 0, \forall i \in \mathcal{S}$, assume that there exists a positive constant $p_i > 0$, such that $z_i(t) \leq p_i$, then the ℓth broadcast period of link j, $B_j[\ell] = T_j^L[\ell+1] - T_j^L[\ell]$, is lower bounded by*

$$B_j[\ell] \geq w_j \ln \left(1 + \frac{\delta_j |\hat{\mu}_j(T_j^L[\ell])|}{|\hat{\mu}_j(T_j^L[\ell])| + \sum_{i \in \mathcal{S}_j} p_i} \right)$$

Proof: Since all the assumptions in corollary 3.3.2 hold, this means link j will obtain a new sample of $\mu_j(t)$ when the inequality in equation 3.40 is about to be violated.

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 error $e_j(t)$ over the time interval. Define $\Omega = \{t \in$

$[T_j^L[\ell], T_j^L[\ell+1]) | |e_j(t)| = 0\}$ On $t \in [T_j^L[\ell], T_j^L[\ell+1]) - \Omega$, we have

$$\begin{aligned}
\frac{d|e_j(t)|}{dt} &\leq \left|\frac{de_j(t)}{dt}\right| \\
&= \left|\frac{d\mu_j(t)}{dt}\right| \\
&= \left|\frac{1}{w_j}\frac{ds_j(t)}{dt} + \frac{1}{w_j}\sum_{i\in\mathcal{S}_j} z_i(t)\right| \\
&= \left|-\frac{1}{w_j}\mu_j(t) + \frac{1}{w_j}\sum_{i\in\mathcal{S}_j} z_i(t)\right| \\
&\leq \frac{1}{w_j}|e_j(t)| + \frac{1}{w_j}|\hat{\mu}_j(T_j^L[\ell])| + \frac{1}{w_j}\sum_{i\in\mathcal{S}_j} p_i \qquad (3.59)
\end{aligned}$$

where we use the right-hand sided derivative when $t = T_j^L[\ell+1]$. We can then solve the differential inequality in equation 3.59, which gives us

$$|e_j(t)| \leq e^{\frac{1}{w_j}(t-T_j^L[\ell])}|e_j(T_j^L[\ell])| + \left(|\hat{\mu}_j(T_j^L[\ell])| + \sum_{i\in\mathcal{S}_j} p_i\right)\left[e^{\frac{1}{w_j}(t-T_j^L[\ell])} - 1\right] \qquad (3.60)$$

for all $t \in [T_j^L[\ell], T_j^L[\ell+1])$ since $|e_j(t)| = 0$ for all $t \in \Omega$.

Since link j's next broadcast will be triggered when the inequality in equation 3.40 is about to be violated. This will happen at time $T_j^L[\ell+1]$ when

$$|e_j(T_j^L[\ell+1])| \geq \delta_j|\hat{\mu}_j(T_j^L[\ell])| \qquad (3.61)$$

We can use the bound on $|e_j(t)|$ in equation 3.60 to solve for $T_j^L[\ell+1]$ in equation

3.61. This gives us a lower bound of the broadcast period of link j

$$\begin{aligned}
B_j[\ell] &= T_j^L[\ell+1] - T_j^L[\ell] \\
&\geq w_j \ln\left(\frac{(\delta_j+1)|\hat{\mu}_j(T_j^L[\ell])| + \sum_{i\in\mathcal{S}_j} p_i}{|e_j(T_j^L[\ell])| + |\hat{\mu}_j(T_j^L[\ell])| + \sum_{i\in\mathcal{S}_j} p_i}\right) \qquad (3.62)\\
&= w_j \ln\left(1 + \frac{\delta_j|\hat{\mu}_j(T_j^L[\ell])| - |e_j(T_j^L[\ell])|}{|e_j(T_j^L[\ell])| + |\hat{\mu}_j(T_j^L[\ell])| + \sum_{i\in\mathcal{S}_j} p_i}\right) \\
&= w_j \ln\left(1 + \frac{\delta_j|\hat{\mu}_j(T_j^L[\ell])|}{|\hat{\mu}_j(T_j^L[\ell])| + \sum_{i\in\mathcal{S}_j} p_i}\right) \qquad (3.63)
\end{aligned}$$

This completes the proof. ■

The above corollary provides a state-dependent lower bound for link j's broadcast period, $B_j[\ell]$. Since p_i is a known constant, then $B_j[\ell]$ is only a function of link j's local state, which means link j could predict when the next broadcast will happen. However, this bound seems to be rather conservative.

3.3.5 Event-triggering with transmission delays

The data dropout analysis in subsection 3.3.3 guarantees the convergence of the algorithm when the feedback information is not 'too out of date'. In this subsection we study the effect of transmission delay on the stability of the algorithm. In particular, we gives an upper bound on the maximum tolerable delay while ensuing the convergence of the event-triggered algorithm.

To study the effect of transmission delay on the stability of the algorithm, let us first see what happens when there are transmission delays in the network. Suppose link j detects a local event and obtains a new sampled state $\hat{\mu}_j(t)$ at time t. Link j then transmits the new $\hat{\mu}_j(t)$ to users $i \in \mathcal{S}_j$, where user i receives the new state after some time $\tau_{ji}(t) > 0$. Here $\tau_{ji}(t)$ denotes the transmission delay from link $j \in \mathcal{L}$ to user $i \in \mathcal{S}_j$ when the broadcasted link state is sampled at time t. It is obvious to see that $\tau_{ji}(t)$ may be different for each user i. We define a maximum delay

$$\tau_j(t) = \max_{i \in \mathcal{S}_j} \tau_{ji}(t) \tag{3.64}$$

In the following analysis, we assume that $\tau_{ji}(t) = \tau_j(t)$, $\forall t > 0$, $\forall j \in \mathcal{L}$, $\forall i \in \mathcal{S}_j$. So our analysis focuses on the worst-case scenario, where each link-user pair experiences the maximum possible transmission delay. It is not difficult to see that, when not all link-user pairs experience the maximum transmission delay, the result of our analysis will apply as well. What's left to determine is the maximum possible $\tau_j(t)$ at time t for link j.

The following theorem gives an upper bound on the maximum allowable delay as a function of the local link state assuming there is no data dropouts.

Theorem 3.3.5 *Suppose all the assumptions in corollary 3.3.2 hold except that link j's ℓth sampled link state $\hat{\mu}_j(T_j^L[\ell])$ is broadcasted to the users $i \in \mathcal{S}_j$ with a nonzero transmission delay $\tau_j(T_j^L[\ell])$. Assume that there is no data dropouts in the system. Also $\forall t \geq 0, \forall i \in \mathcal{S}$, assume that there exists a positive constant $p_i > 0$, such that $z_i(t) \leq p_i$. Assume the transmission delay satisfies*

$$\tau_j(T_j^L[\ell]) \leq w_j \ln\left(1 + \frac{\frac{1}{\sqrt{\frac{1}{2}\overline{LS}}}(1-\delta_j)^{k_j+1}\left|\hat{\mu}_j(T_j^L[\ell])\right|}{\left|\hat{\mu}_j(T_j^L[\ell])\right| + \sum_{i \in \mathcal{S}_j} p_i}\right)$$

where $k_j = 0, 1, 2, \cdots$ is the number of link j's broadcast between the broadcast and successful receipt of the ℓth sampled state $\hat{\mu}_j(T_j^L[\ell])$, then the user rates $x(t)$ asymptotically converge to the unique minimizer of $L(x, s; \lambda, w)$.

Proof: Consider link $j \in \mathcal{L}$, it samples its state for the ℓth time at time $T_j^L[\ell]$, and then broadcasts the sampled state $\hat{\mu}_j(T_j^L[\ell])$ to users $i \in \mathcal{S}_j$. The packet will experience a delay of $\tau_j(T_j^L[\ell])$. For notation convenience, we simply use τ_j in the proof here. Consider the time interval $t \in [T_j^L[\ell], T_j^L[\ell] + \tau_j)$, it is possible that link j will have additional broadcasts besides the broadcast at time $t = T_j^L[\ell]$. Suppose the number of additional broadcast is k_j, where $k_j = 0, 1, 2, \cdots$.

Define error as $e_j(t) = \mu_j(t) - \hat{\mu}_j(T_j^L[\ell])$ on $t \in [T_j^L[\ell], T_j^L[\ell] + \tau_j)$. We can study the behavior of the error $e_j(t)$ over the time interval. Similar to the proof of corollary 3.3.4, we can show that

$$|e_j(t)| \leq e^{\frac{1}{w_j}(t - T_j^L[\ell])}|e_j(T_j^L[\ell])| + \left(|\hat{\mu}_j(T_j^L[\ell])| + \sum_{i \in \mathcal{S}_j} p_i\right)\left[e^{\frac{1}{w_j}(t - T_j^L[\ell])} - 1\right] \quad (3.65)$$

for all $t \in [T_j^L[\ell], T_j^L[\ell] + \tau_j)$.

From equation 3.40, we know

$$|\mu_j(t)| \geq (1 - \delta_j)|\hat{\mu}_j(T_j^L[\ell + k])| \quad (3.66)$$

for all $t \in [T_j^L[\ell + k], T_j^L[\ell] + \tau_j)$. Apply the above equation repeatedly on $[T_j^L[\ell + k - 1], T_j^L[\ell + k])$,$\cdots$,$[T_j^L[\ell], T_j^L[\ell + 1])$, we know

$$|\mu_j(t)| \geq (1 - \delta_j)^{k_j + 1}|\hat{\mu}_j(T_j^L[\ell])| \quad (3.67)$$

for all $t \in [T_j^L[\ell + k], T_j^L[\ell] + \tau_j)$.

If we can ensure

$$|e_j(t)| \leq \frac{1}{\sqrt{\frac{1}{2}\overline{LS}}}|\mu_j(t)| \quad (3.68)$$

on $t \in [T_j^L[\ell+k], T_j^L[\ell]+\tau_j)$, and since we have assumed that the positive projection on μ_j is not active, then from theorem 3.3.1, we know that the system will be asymptotic stable. This can be achieved by ensuring

$$e^{\frac{1}{w_j}(t-T_j^L[\ell])}|e_j(T_j^L[\ell])| + \left(|\hat{\mu}_j(T_j^L[\ell])| + \sum_{i\in\mathcal{S}_j} p_i\right)\left[e^{\frac{1}{w_j}(t-T_j^L[\ell])} - 1\right] \\ \leq \frac{1}{\sqrt{\frac{1}{2}\overline{LS}}}(1-\delta_j)^{k_j+1}|\hat{\mu}_j(T_j^L[\ell])| \qquad (3.69)$$

Remember $e_j(T_j^L[\ell]) = 0$ and solve the above inequality, we get

$$t - T_j^L[\ell] \ \leq \ w_j \ln\left(1 + \frac{\frac{1}{\sqrt{\frac{1}{2}\overline{LS}}}(1-\delta_j)^{k_j+1}\left|\hat{\mu}_j(T_j^L[\ell])\right|}{\left|\hat{\mu}_j(T_j^L[\ell])\right| + \sum_{i\in\mathcal{S}_j} p_i}\right)$$

This means if

$$\tau_j(T_j^L[\ell]) \ \leq \ w_j \ln\left(1 + \frac{\frac{1}{\sqrt{\frac{1}{2}\overline{LS}}}(1-\delta_j)^{k_j+1}\left|\hat{\mu}_j(T_j^L[\ell])\right|}{\left|\hat{\mu}_j(T_j^L[\ell])\right| + \sum_{i\in\mathcal{S}_j} p_i}\right)$$

then asymptotic stability of the system is guaranteed. ■

The above theorem gives a state-dependent upper bound for the maximum allowable delay of the broadcasted state. However, the upper bound is also a function of an unknown variable k_j, which is somewhat undesirable. An upper bound or exact form of k_j turns out to be difficult to obtain. However, we can show that under certain conditions, we can have an estimate of k_j's upper bound.

Corollary 3.3.6 *In theorem 3.3.5, if*

$$|\hat{\mu}_j(T_j^L[\ell])| \ll \sum_{i\in\mathcal{S}_j} p_i \quad \textit{and} \quad \overline{LS} \gg 2\rho_j \qquad (3.70)$$

for all $j \in \mathcal{S}$ and all $\ell = 0, 1, \cdots$, then an upper bound of k_j can be approximated by $\frac{1}{\sqrt{\rho_j}}$.

Proof: Recall that if $x \leq 1$, the Taylor series of $\ln(1+x)$ can be written as

$$\ln(1+x) = \sum_{n=1}^{\infty} (-1)^{n+1} \frac{x^n}{n} \tag{3.71}$$

This means if $x \ll 1$, we have $\ln(1+x) \approx x$.

If we combine the result in theorem 3.3.5 on transmission delay $\tau_j(T_j^L[\ell])$ and the result in corollary 3.3.4 on broadcast period $B_j[\ell]$, we know that an upper bound of k_j is approximately given by

$$\begin{aligned}
k_j &\approx \frac{\tau_j(T_j^L[\ell])}{B_j[\ell]} && (3.72)\\
&\leq \left(\frac{\frac{1}{\sqrt{\frac{1}{2}\overline{LS}}}(1-\delta_j)^{k_j+1} \left| \hat{\mu}_j(T_j^L[\ell]) \right|}{\left| \hat{\mu}_j(T_j^L[\ell]) \right| + \sum_{i \in \mathcal{S}_j} p_i} \right) \left(\frac{|\hat{\mu}_j(T_j^L[\ell])| + \sum_{i \in \mathcal{S}_j} p_i}{\delta_j |\hat{\mu}_j(T_j^L[\ell])|} \right) && (3.73)\\
&= \frac{1}{\sqrt{\frac{1}{2}\overline{LS}}}(1-\delta_j)^{k_j+1} \frac{1}{\delta_j} && (3.74)\\
&= \left(\frac{\sqrt{\frac{1}{2}\overline{LS}}}{\sqrt{\frac{1}{2}\overline{LS}} + \sqrt{\rho_j}} \right)^{k_j} \frac{1}{\sqrt{\rho_j}} && (3.75)\\
&\approx \frac{1}{\sqrt{\rho_j}} && (3.76)
\end{aligned}$$

Here the first approximation follows easily. The first inequality follows by using the first assumption in equation 3.70 and the fact that $\ln(1+x) \approx x$. The two equalities follow by using applying the definition of δ_j. The last approximation follows by using the $\overline{LS} \gg 2\rho_j$ assumption. ■

The corollary basically says that if ρ_j is small, then we should expect a large k_j, and vice versa. This certainly makes sense since when ρ_j is small, the link broadcasts more often, and as a result we would expect a larger k_j. The two assumptions in equation 3.70 are reasonable. The first assumption holds since our estimate of p_i is usually rather conservative, and the second assumption holds since $\rho_j \leq 1$, and $\overline{LS} \gg 1$ holds for most real-life networks.

3.4 Simulation

This section presents simulation results. We compare the number of message exchanges of our two event-triggered algorithms against the dual decomposition algorithm. Simulation results show that both event-triggered algorithms reduce the number of message exchanges by up to two orders of magnitude when compared to dual decomposition. Moreover, our event-triggered algorithms are scale free with respect to network size. The robustness to data dropouts and transmission delays of the event-triggered primal-dual algorithm are also demonstrated.

The remainder of this section is organized as follows: Subsection 3.4.1 discusses the simulation setup. Simulation results on broadcast periods of both event-triggered algorithms are shown in subsection 3.4.2. Subsection 3.4.3 and 3.4.4 present simulation results of the event-triggered primal-dual algorithm when data dropouts and transmission delays are taken into considerations respectively. The scalability results with respect to $\overline{S}$ and $\overline{L}$ are presented in subsection 3.4.5 and 3.4.6, respectively.

3.4.1 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 a utility function $U_i(x_i) = \alpha_i \log x_i$ to 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, all three algorithms are simulated. The optimal rate x^* and its corresponding utility U^* are calculated using a global optimization technique.

Define error as (for all algorithms)

$$e(k) = \left| \frac{U(x(k)) - U^*}{U^*} \right| \tag{3.77}$$

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 all 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. We should point out that since we are comparing a primal algorithm and a primal-dual algorithm (our event-triggered algorithms) with a dual algorithm, and they run at different time-scales, iteration number is then a more appropriate measure of convergence than time [15] [26].

The default settings for simulation are as follows: $M = 60$, $N = 150$, $\overline{L} = 8$, $\overline{S} = 15$, $e_d = 3\%$. For all three 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 1.14. In our event-triggered primal algorithm, $\rho = 0.5$, $\lambda_j = 0$, $w_j = 0.01$ for $j \in \mathcal{L}$. In the event-triggered primal-dual algorithm, $\rho_j = 0.9$, $\lambda_j = 0$, $w_j = 0.01$ for $j \in \mathcal{L}$.

3.4.2 Broadcast periods of the event-triggered algorithm

In this subsection we present simulation results on the broadcast periods of our event-triggered algorithms. Simulation shows that both event-triggered algorithms have much longer average broadcast period than dual decomposition.

This simulation uses the default settings in subsection 3.4.1 and ran for 3.60s. The error in all three algorithms entered the e_d neighborhood in $3.60s$. For reference, with the same settings, the average broadcast period for the dual decomposition is 0.0026. In our event-triggered primal algorithm, there are a total of 295 link events, and 1059 user events. So the links have an average broadcast period of 0.7332, which is 282 times longer than in dual decomposition. However, unlike in dual decomposition, the users in the primal algorithm have to broadcast their states occasionally as well, and their average broadcast period is 0.5099. For the event-triggered primal-dual algorithm, there are a total of 2106 link events. The links have an average broadcast period of 0.1026, which is 39 times longer than in dual decomposition. As we can easily see, the primal algorithm has the longest average link broadcast period among three algorithms. However, the algorithm need users to broadcast their states as well, which does not happen in the primal-dual algorithm and dual decomposition. Both the primal and primal-dual algorithms enjoy much longer broadcast periods than dual decomposition.

To see how the broadcast periods vary for a particular user or link in our event-triggered algorithms, we pick one user and one link in the network, and

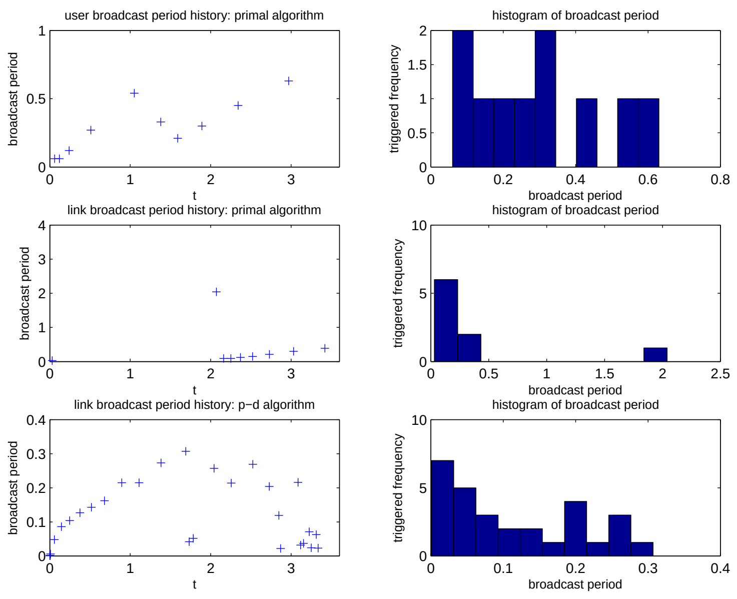

Figure 3.1. Broadcast results for event-triggered algorithms

their broadcast results are shown in figure 3.1. The four plots above correspond to the user and link broadcast in the primal algorithm, and the two plots below correspond to the link broadcast in the primal-dual algorithm. The top left plot in figure 3.1 is the time history of broadcast periods generated by the user's local event in the primal algorithm. The top right plot is the histogram of those user broadcast periods. This user's broadcast periods range between 0.0600 and 0.6300, with the minimum broadcast period 23 times longer than in dual decomposition. This user was triggered 10 times, with an average broadcast period of 0.3600. The middle left plot is the time history of broadcast periods generated by the link's local event in the primal algorithm. The middle right plot is the histogram of those link broadcast periods. This link's broadcast periods range between 0.0300 and 2.0400, with the minimum broadcast period 11 times longer than in dual decomposition. This link was triggered 9 times, with an average broadcast period of 0.4000. The above broadcast period results was generated for an active link. For all the inactive links, it is interesting to see that they never broadcasted in the primal algorithm. This means only active links need to send their feedback signals to the users in the primal algorithm.

Finally, the lower left plot is time history of broadcast periods generated by the link's local event in the primal-dual algorithm. The lower right plot is the histogram of those link broadcast periods. The link's broadcast periods range between 0.0020 and 0.3070. The link was triggered 30 times, with an average broadcast period of 0.1150. Unlike in the primal algorithm, the inactive links also need to send their feedback signals to the users in the primal-dual algorithm.

Based on the above discussion, we know that in the primal algorithm, the users and active links need to broadcast their states. In the primal-dual algorithm and dual decomposition, all links need to broadcast their states. Despite different broadcast strategies, both event-triggered algorithms generate much longer broadcast periods than dual decomposition.

3.4.3 Data dropout simulation

In this subsection we present simulation results on the primal-dual algorithm when data dropouts are taken into consideration.

From the discussion of the data dropout in subsection 3.3.3, we know that each link j's largest number of successive data dropout d_j needs to satisfy the inequality in equation 3.50 to ensure stability. Given a network, the max allowable number of successive data dropouts $D_j(\rho_j)$ is a monotone decreasing function 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 $\rho = 0.9, 0.218, 0.102, 0.025$, $D_j(\rho_j) = 0, 1.0034, 2.0052, 5.0089$ respectively. Table 3.1 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 are always equal to the largest number of successive data dropouts, d_j. When $\rho_j = 0.102$ and $\rho_j = 0.025$, d_j is chosen so that the stability condition in equation 3.50 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 3.50 is indeed a sufficient condition for stability, but is a rather conservative bound. Remember in subsection 3.4.2, when $\rho_j = 0.9$ and no data dropouts, the average broadcast period is 0.1096. This is very close to the average successful broadcasts period in table 3.1 in all three cases. However, with no surprise, when data dropouts occurs, the average triggered broadcasts period is much shorter than 0.1096, which is clearly shown in table 3.1.

3.4.4 Transmission delay simulation

In this subsection we present simulation results on the primal-dual algorithm when transmission delays are taken into consideration. Simulation shows that the algorithm still converges to the equilibrium when the delay is not too long.

We consider the scenario that when each $\hat{\mu}_j$ packet is broadcasted, it experiences a uniform random transmission delay ranging from 0ms to τms, and we choose different values of τ in the simulation. To be specific, τ is chosen to be 0ms, 10ms, 20ms, 50ms, 100ms, respectively. Note that here in the simulation, the transmission delay of each link-user pair can be different and is independent

ρ_j	0.102	0.025	0.9
$D_j(\rho_j)$	2.0052	5.0089	0
d_j	2	5	10
Number of triggered broadcasts	7490	18336	27765
Number of successful broadcasts	2474	3029	2497
Average triggered broadcasts period	0.0288	0.0118	0.0078
Average successful broadcasts period	0.0873	0.0713	0.0865

TABLE 3.1

Simulation results for different ρ_j and d_j

of each other. The user always uses the latest received $\hat{\mu}_j$ packet regardless of the sending time. The simulation uses the default settings in subsection 3.4.1 and ran for 3.60s. The error in all three algorithms entered the e_d neighborhood in 3.60s. We plot the error $e(k)$ as a function of the iteration number k for different τ in figure 3.2. From bottom to top, the five plots correspond to $\tau = 0$ms, 10ms, 20ms, 50ms, 100ms, respectively. As we can see, the error enters the e_d neighborhood in all five scenarios. Of course, when τ increase, the iteration number k it takes to converge increases as well. For each different τ, iteration number $k = 33, 35, 41, 105, 148$, respectively. The increase in iteration number is expected with the use of delayed information. This simulation clearly demonstrated the effectiveness of our event-triggered algorithm with the presence of transmission delay.

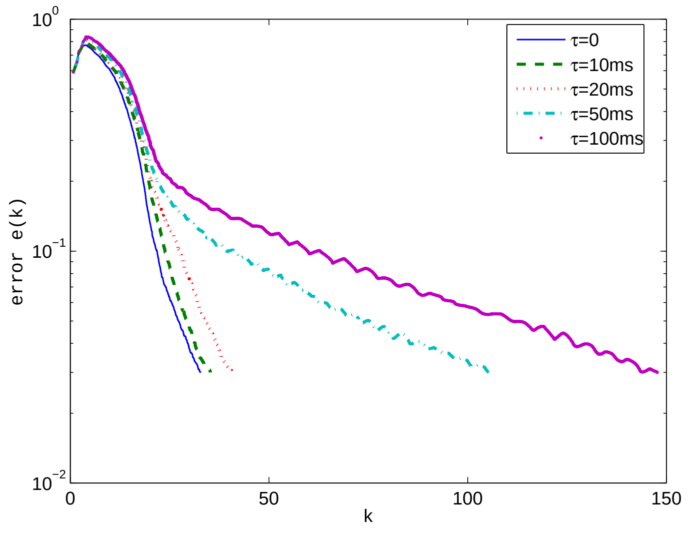

Figure 3.2. Error $e(k)$ as a function of iteration number k for different τ

3.4.5 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 $\overline{S}$, all algorithms were run 1500 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 three algorithms. Figure 3.3 plots the iteration number K (in logarithm scale) as a function of $\overline{S}$ for all algorithms. The asterisks above represent m_K for dual decomposition, the crosses in the middle denote our primal-dual algorithm, while the circles below correspond to the primal algorithm. The dotted vertical line around each asterisk, cross and circle corresponds to the interval $[m_K - \sigma_K, m_K + \sigma_K]$ for each different $\overline{S}$ denoted by the x-axis.

For both the primal and primal-dual algorithm, when $\overline{S}$ increases from 7 to 26, m_K does not show noticeable increase. For the primal algorithm, m_K varies between 15.1 and 21.1, and σ_K varies between 1.1 and 1.6. For the primal-dual

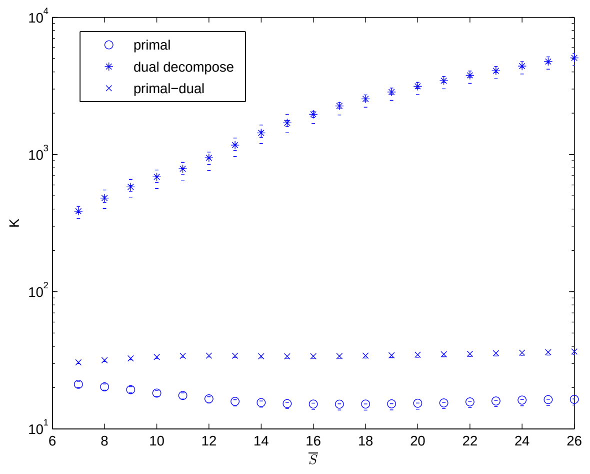

Figure 3.3. Iteration number K as a function of $\overline{S}$ for all algorithms.

algorithm, m_K increases from 30.5 to 36.6, and σ_K increases from 0.7 and 1.9. As for dual decomposition, m_K increases from 0.3856×10^3 to 5.0692×10^3. σ_K at the same time increases from 0.4695×10^2 to 6.4627×10^2. Our event-triggered algorithms are up to two orders of magnitude faster than the dual decomposition. We can also see that, unlike the dual decomposition algorithm, which scales superlinearly with respect to $\overline{S}$, both the primal and primal-dual event-triggered algorithms on the other hand are scale-free.

3.4.6 Scalability with respect to $\overline{L}$

This simulation is similar to subsection 3.4.5 except that we vary $\overline{L}$ from 4 to 18 instead of $\overline{S}$. Figure 3.4 plots K (in logarithm scale) as a function of $\overline{L}$ for all algorithms.

For both the primal and primal-dual algorithm, when $\overline{L}$ increases from 4 to 18, m_K increases slowly. For the primal algorithm, m_k increases from 15.0 to 18.2, and σ_K varies between 1.0 and 1.4. For the primal-dual algorithm, m_K increases from 31.1 to 48.5, and σ_K varies between 1.1 and 3.8. As for dual decomposition, m_K increases from 0.9866×10^3 to 3.5001×10^3, and σ_K at the same time increases from 0.9991×10^2 to 6.0232×10^2. Our event-triggered algorithms again are up to two orders of 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 algorithms on the other hand are scale-free.

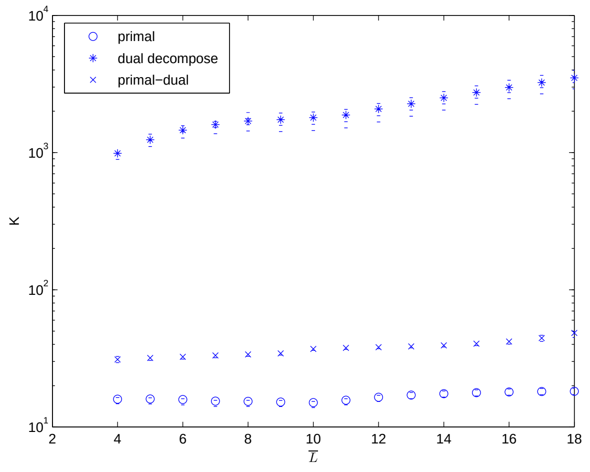

Figure 3.4. Iteration number K as a function of $\overline{L}$ for all algorithms.

3.5 Conclusion

This chapter presents a primal and a primal-dual event-triggered distributed NUM algorithms based on the augmented Lagrangian methods. The chapter establishes state-dependent event-triggering thresholds under which the proposed algorithms converge to the approximate solution of the NUM problem. For the primal-dual algorithm, bounds on the maximum allowable data dropouts and maximum allowable transmission delays are given. A state-dependent lower bound on the broadcast period is presented as well. Simulation results suggest that both algorithms are scale-free with respect to two measures of network size, and reduce the number of message exchanges by up to two orders of magnitude when compared to existing dual decomposition algorithms.

CHAPTER 4

Optimal power flow in microgrids using event-triggered optimization

This chapter uses the optimal power flow problem in microgrids as a nontrivial real-life example to demonstrate the effectiveness of event-triggered optimization.

4.1 Introduction

Microgrids [33] [22] [18] are power generation/distribution systems in which users and generators are in close proximity. This results in relatively low voltage grids (few hundred kVA). Generation is often done using renewable generation sources such as photovoltaic cells or wind turbines. Power generation can also be accomplished through small microturbines and gas/diesel generators. Storage devices such as battery banks represent another important power source for microgrids. These units can be used in places such as office buildings, parks, homes and battle fields as distributed power sources. They are modular in the sense that, if needed, new unit can be added to the network in an easy way. All the microgrids in the network can work in a cooperative way to meet the overal load demand in the network.

Active/reactive power dispatch problems have been the research subject of power system community since in the early 1960's. The problem is usually formulated as an optimal power flow (OPF) problem. The OPF problem [40] [20] is an important class of problems in the power industry. The problem is to determine generator power set points so that the overall cost of power generation is minimized, while respecting limits on the generator's capacity and transmission power flow constraints. This chapter examines the optimal active power flow problem [40], which we will simply denote as the OPF problem in the remainder of the chapter.

Various centralized or distributed optimization algorithms have been proposed to solve the OPF problem, including network flow approach [11], interior point method [40] [20], multi-area decomposition [17] [9] [29], etc. These algorithms usually made the assumptions that communication between subsystems was not expensive and reliable. This assumption, however, is not always realistic since one of the first things to go down during power disruptions is the communication network. One way around this problem is to make use of low power ad hoc wireless communication networks that operate independently of the main power grid.

Ad hoc wireless sensor networks have recently been used in the reliable operation and control of various civil infrastructure systems [45][37]. The nodes in these networks are usually powered by batteries or solar arrays, so they would be unaffected by fluctuations in the main power grid. These networks, however, have severe throughput limitations that make it impractical to send a large amount of information across the network. Moreover, it may be impractical to send periodically sampled data across the network as the nodes in these networks have limited power due to their reliance on batteries. As a result of these considerations, ad hoc communication networks can provide a power grid's communication infrastructure only if we can greatly limit the amount of information that needs to be transmitted across the network. One way of doing this is to adopt an event-triggered approach to information transmission.

In event-triggered systems, sensors are sampled in a *sporadic* non-periodic manner. Our work in chapter 2 and 3 has shown that event-triggering can greatly reduce (by several orders of magnitude) the message passing required in the solution of network utility maximization problems. Event-triggering therefore may provide a useful approach for reducing the power grid's use of the communication network.

The OPF problem is similar to the NUM problem we considered in previous chapters. In this chapter, we develop an event-triggered distributed optimization algorithm for the OPF problem and prove its convergence. We use the CERTS microgrid model [32] as an example power system to show the effectiveness of our algorithm. The simulation is done in MATLAB/SimPower and shows that our algorithm solves the OPF problem in a distributed way, and the communication between neighboring subsystems is very infrequent. The rest of the chapter is organized as follows. Section 4.2 formally states the OPF problem. Section 4.3 briefly introduces the main components and mechanisms of the CERTS microgrid model. Section 4.4 presents an event-triggered scheme to solve the OPF in a distributed way. Simulation and conclusions are in section 4.5 and section 4.6, respectively.

4.2 The optimal power flow (OPF) problem

This section derives the DC flow model [16], which is widely used to characterize a power system's behavior around the normal steady state operation. It then uses the DC flow model to formally state the OPF problem, which we will solve later in section 4.4.

The power system can be modeled as a directed graph, which is shown in figure 4.1. Consider a connected directed graph $G = (\mathcal{V}, \mathcal{E})$ as an abstraction of an electrical power network. The system consists of N buses. For simplicity of discussion, we assume that each bus has a local generator and a local load connected to it in our model. More general scenarios where a certain bus does not have a local generator or has multiple local loads can be treated similarly without much difficulty. $\mathcal{V} = \{v_1, ..., v_N\}$ is the set of nodes, and each node represents a bus (with a local generator and load). $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of directed edges, which corresponds to the power transmission lines. Suppose the network has $M = |\mathcal{E}|$ edges, and they are ordered $1, 2, ..., M$. An edge from node i to node j is denoted as $e_{ij} = (v_i, v_j)$. $z_{ij} = r_{ij} + jx_{ij}$ is the impedance of the transmission line corresponding to edge e_{ij}. Since r_{ij} is often negligible compared to x_{ij}, we can assume $r_{ij} = 0$ in our DC flow model. Suppose the incidence matrix [10] of graph G is I, and define a diagonal matrix $D \in \mathbb{R}^{M \times M}$, whose diagonal entries are the reactances x of M transmission lines. Then the weighted incidence matrix $A \in \mathbb{R}^{M \times N}$ is defined as $A = DI$. The set of neighbors of node i is defined as $\mathcal{N}(i) = \{v_j \in \mathcal{V} | (v_i, v_j) \in \mathcal{E}\}$, and node i has $|\mathcal{N}(i)|$ neighbors. The set of transmission lines that leave bus i is defined as $\mathcal{L}(i) = \{e_{ij} \in \mathcal{E} | j \in \mathcal{N}(i)\}$.

In the power system analysis, complex powers are extensively used [7] [30]. Remember power can be defined as $P = Re\{ui^*\}$, where u and i represents the voltage and current, respectively. $Re\{a\}$ and $Im\{a\}$ is the real and imaginary part of a complex number a. Complex power S and reactive power Q are defined as

$$S = ui^*, \quad Q = Im\{ui^*\} \tag{4.1}$$

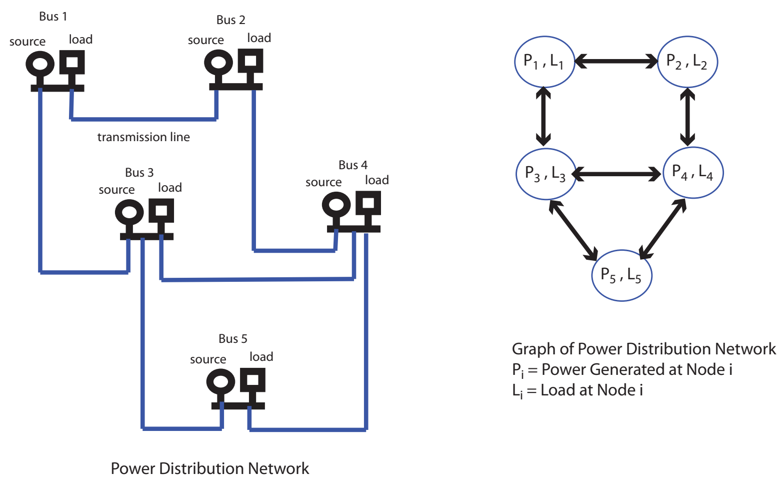

Figure 4.1. Power distribution network and graph abstract

It is trivial to see $S = P + jQ$.

Let S_{ij} denote the complex power flow from node i to node j, and u_i denote the generator voltage at node i. Use the following magnitude-phase representation,

$$u_i = |u_i|e^{j\theta_i}, \quad u_j = |u_j|e^{j\theta_j} \tag{4.2}$$

and remember $z_{ij} = jx_{ij}$, we have

$$S_{ij} = u_i(\frac{u_i - u_j}{z_{ij}})^* = P_{ij} + jQ_{ij} \tag{4.3}$$

$$S_{ji} = u_j(\frac{u_j - u_i}{z_{ji}})^* = P_{ji} + jQ_{ji} \tag{4.4}$$

where

$$P_{ij} = -P_{ji} = \frac{|u_i||u_j|}{x_{ij}} \sin(\theta_i - \theta_j) \tag{4.5}$$

$$Q_{ij} = \frac{|u_i|^2}{x_{ij}} - \frac{|u_i||u_j|}{x_{ij}} \cos(\theta_i - \theta_j) \tag{4.6}$$

$$Q_{ji} = \frac{|u_j|^2}{x_{ji}} - \frac{|u_j||u_i|}{x_{ji}} \cos(\theta_i - \theta_j) \tag{4.7}$$

Under normal operating conditions, we have $|u_i| \approx |u_j|$, and $\theta_i - \theta_j$ is typically small. In this case, there is reasonably good decoupling between the control of active power flow P_{ij}, P_{ji} and reactive power flow Q_{ij}, Q_{ji}. The active power flow is mainly dependent on $\theta_i - \theta_j$, and the reactive power flow is mainly dependent on $|u_i| - |u_j|$ [7].

The DC flow model we are using further assumes that only the voltage phases θ_i, θ_j vary, and that variation is small. Voltage magnitudes $|u_i|$, $|u_j|$ are assumed to be constant ($|u_i| = |u_j| = 1$ here). In this case, the reactive power flow Q_{ij} is negligible, and we are only considering the active power flow P_{ij}. With the assumptions and simplifications above, the power flow from node i to node j is given by

$$P_{ij} = \frac{1}{x_{ij}}(\theta_i - \theta_j) \tag{4.8}$$

The total power flowing into bus i, P_i, must equal the power generated by generator i minus the power absorbed by the local load at the bus. P_i, therefore, must equal the the sum of the power flowing away from bus i on all transmission lines. This means

$$P_i = \sum_{j \in \mathcal{N}(i)} P_{ij} = \sum_{j \in \mathcal{N}(i)} \frac{1}{x_{ij}} (\theta_i - \theta_j) \tag{4.9}$$

which can be expressed in a matrix form

$$P = B\theta \tag{4.10}$$

where $P = [P_1, ..., P_N]^T$, $\theta = [\theta_1, ..., \theta_N]^T$, and B is defined as

$$B_{ij} = \begin{cases} \sum_{j \in \mathcal{N}(i)} \frac{1}{x_{ij}}, & \text{if} \quad i = j, \\ -\frac{1}{x_{ij}}, & \text{if} \quad e_{ij} \in E, \\ 0, & \text{if} \quad e_{ij} \notin E \end{cases} \tag{4.11}$$

B is a singular matrix, which can be thought as a weighted Laplacian matrix [10] of the graph. The weight here is $1/x_{ij}$ for edge e_{ij}.

Based on our DC flow model in equation 4.8, we can formulate the **General OPF problem** as follows :

$$\begin{aligned} \text{minimize} \quad & C(P_G) = \textstyle\sum_{i=1}^{N} C_i(P_{G_i}) && (4.12) \\ \text{w.r.t} \quad & P_G && (4.13) \\ \text{subject to:} \quad & B\theta = P_G - P_L && (4.14) \\ & \underline{P_G} \leq P_G \leq \overline{P_G} && (4.15) \\ & \underline{P} \leq A\theta \leq \overline{P} && (4.16) \end{aligned}$$

Here $P_G = [P_{G_1}, ..., P_{G_N}]^T$ is the vector of generated active powers for all generators, and $P_L = [P_{L_1}, ..., P_{L_N}]^T$ is the vector of total local loads for all buses. A and B are sparse matrices and have been defined previously. $\underline{P_G} = \{\underline{P_{G_1}}, \cdots, \underline{P_{G_N}}\}$ and $\overline{P_G} = \{\overline{P_{G_1}}, \cdots, \overline{P_{G_N}}\}$ represent the lower and upper limits of the generators' power generating constraints. $\underline{P} = \{\underline{P_1}, \cdots, \underline{P_M}\}$ and $\overline{P} = \{\overline{P_1}, \cdots, \overline{P_M}\}$ represent the lower and upper limits of the power flows on the transmission lines. Here P_L, $\underline{P_G}$, $\overline{P_G}$, $\underline{P}$ and $\overline{P}$ are known constants in the problem formulation, and A and B are known constant matrices. The objective function in equation 4.12 represents the total generation cost of all the generators, and generator i's cost of generating P_{G_i} unit of active power is usually in the form of

$$C_i(P_{G_i}) = a_i + b_i P_{G_i} + c_i P_{G_i}^2 \tag{4.17}$$

where a_i, b_i and c_i are constant coefficients. The constraint in equation 4.14 is the power flow balance equation. Constraints in equation 4.15 and 4.16 represent the generation limits of the generators, and power flow limits on the transmission lines, respectively. The General OPF problem seeks to find the optimal generated active power P_G such that the total generation cost is minimized, subject to the power flow equation and physical constraints of the generation and transmission systems.

To apply the idea of event-triggered optimization, we need to reformulate the previous General OPF problem to fit into the NUM problem formulation and adopt a similar approach we have used in chapter 3. This is done by recognizing that the constraint in equation 4.15 is a power balance relation that is always maintained within the system. We can therefore remove P_G as a control variable to obtain the following **revised OPF problem**.

$$\text{minimize} \quad C(P_G) = \textstyle\sum_{i=1}^{N} C_i((B\theta)_i + P_{L_i}) \tag{4.18}$$

$$\text{w.r.t} \quad \theta \tag{4.19}$$

$$\text{subject to:} \quad \underline{P_G} - P_L \leq B\theta \leq \overline{P_G} - P_L \tag{4.20}$$

$$\underline{P} \leq A\theta \leq \overline{P} \tag{4.21}$$

where $(B\theta)_i$ is the ith element of $B\theta$. Note that the new optimization problem is solved with respect to the phase angle θ instead of P_G. The revised OPF problem also has the same solution as the General OPF problem. We will solve the revised OPF problem later in section 4.4 using an event-triggered distributed algorithm.

4.3 The CERTS Microgrid model

This section briefly describes the CERTS microgrid model and the microsource controller developed by the University of Wisconsin, Madison (UWM). A detailed account of the microsource controllers can be found in [32]. Since we are using the CERTS microgrid model in our MATLAB/SimPower simulation, the basic operation of the model and the controller is described below.

The inverter-based microsource consists of a D.C. source whose outputs are transformed into an A.C. voltage through an inverter. The actions of the inverter are guided by a controller that uses sensed feeder currents and voltages to determine how best to control the operation of the inverter. Figure 4.2 shows that the output of the inverter is passed through a low pass filter to remove switching transients to produce a three phase 480V voltage. A transformer then steps this down to 208 V (120 volts rms).

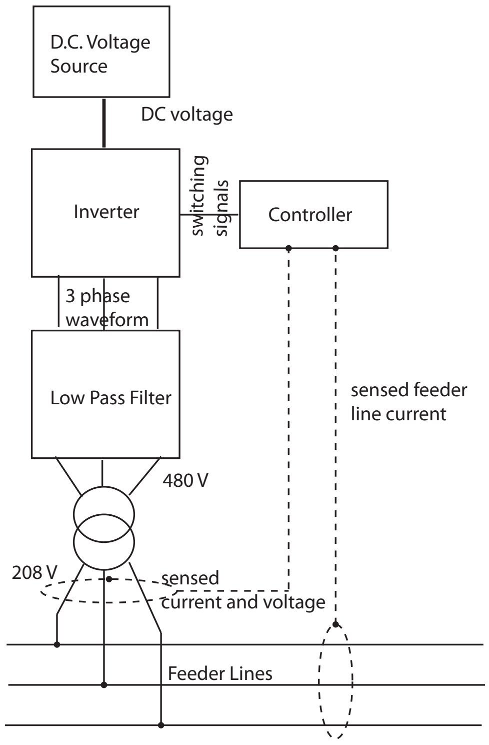

Figure 4.2. Inverter-based microsource

The UWM microsource controller is shown in figure 4.3. The inputs are measurements of inverter current, load voltage and line current. The controller also takes as reference inputs the requested voltage level E_{req}, the requested power set point P_{req}, and the desired frequency f_{req} (usually 60 Hz). The controller takes the measured inputs and computes the instantaneous reactive power, Q, the voltage magnitude, E, and the real power P. These computed values are low pass filtered. The reactive power, Q, and the requested voltage E_{req} are input to the Q vs E droop controller to determine the desired voltage level. This is compared against the measured voltage level and the output V is then given to the gate pulse generator. Another channel in the controller uses the measured real power and implements another droop control that balances the system's frequency against the requested power level, P_{req}. The output of the P vs frequency droop controller is used to adjust the phase angle, δ_V, which is also fed into the gate pulse generator. The output of the gate pulse generator goes directly into the inverter.

The action of this controller is, essentially, to mimic the droop controls seen in traditional synchronous generators. This means that if a load begins drawing a great deal of real power, then the line frequency will "droop" as an indicator of the extra stress on the system. The controller automatically tries to restore that frequency to its desired levels. But it will be unable to restore the droop if the power being pulled if the power drawn by the load exceeds the generator's capacity. This drop in frequency can be sensed at the load and may be used to help decide if the load should disconnect from the microgrid. A similar scenario occurs if the load begins drawing too much reactive power. In this case, there will be a droop in the voltage that can again be used by the load to determine if it should disconnect from the grid. These droop controllers are well understood and they can be easily interfaced to price-based power control methods through the requested power and voltage set points shown in figure 4.3. The event-triggered control inputs developed in this chapter will interface to this controller through the requested power input.

4.4 Event-triggered distributed optimization for OPF

This section develops an event-triggered distributed algorithm to solve the revised OPF problem in section 4.2, and proves its convergence. The event-triggered algorithm can be easily integrated into the CERTS microgrid model by dynamically adjusting the power set point of each generator.

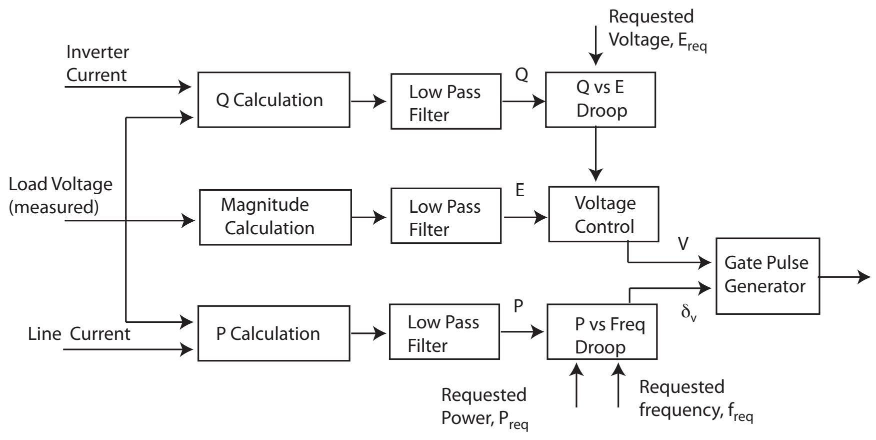

Figure 4.3. UWM microsource controller

The revised OPF problem is a constrained problem, which can be converted into a sequence of unconstrained problems by adding to the cost function a penalty term that prescribes a high cost to infeasible points.

Take the $A\theta \leq \overline{P}$ constraint for example, we can introduce a slack variable $s \in \mathbb{R}^M$ and replace the inequalities $\overline{P_j} - a_j^T\theta \geq 0$, $\forall j \in \mathcal{E}$ by

$$a_j^T\theta - \overline{P_j} + s_j = 0, \quad s_j \geq 0, \quad \forall j \in \mathcal{E} \tag{4.22}$$

Here the vector $a_j^T = [A_{j1}, \cdots, A_{jN}]$ is the jth row of matrix A.

Define

$$\overline{\psi_j}(\theta; w^{\mathcal{E}}) = \min_{s_j \geq 0} \frac{1}{2w_j^{\mathcal{E}}}(a_j^T\theta - \overline{P_j} + s_j)^2 \tag{4.23}$$

where a penalty parameter $w_j^{\mathcal{E}}$ is associated with each transmission line j and $w^{\mathcal{E}} = \left[w_1^{\mathcal{E}}, \cdots, w_M^{\mathcal{E}}\right]$ is the vector of transmission line penalty parameters.

It is easy to show that

$$\overline{\psi_j}(\theta; w^{\mathcal{E}}) = \begin{cases} 0, & \text{if} \quad \overline{P_j} - a_j^T\theta \geq 0 \\ \frac{1}{2w_j^{\mathcal{E}}}(a_j^T\theta - \overline{P_j})^2, & \text{otherwise} \end{cases}$$

Similarly we can define

$$\underline{\psi_j}(\theta; w^{\mathcal{E}}) = \begin{cases} 0, & \text{if} \quad \underline{P_j} - a_j^T\theta \leq 0 \\ \frac{1}{2w_j^{\mathcal{E}}}(a_j^T\theta - \underline{P_j})^2, & \text{otherwise} \end{cases}$$

which corresponds to the inequality constraint $\underline{P_j} - a_j^T\theta \leq 0$, $\forall j \in \mathcal{E}$.

Define $b_k^T = [B_{k1}, \cdots, B_{kN}]$ as the kth row of matrix B and let $w^{\mathcal{V}} = \left[w_1^{\mathcal{V}}, \cdots, w_N^{\mathcal{V}}\right]$ denote the vector of generator penalty parameters. This gives us

$$\overline{\chi_k}(\theta; w^{\mathcal{V}}) = \begin{cases} 0, \quad \text{if} \quad \overline{P_{G_k}} - P_{L_k} - b_k^T\theta \geq 0 \\ \frac{1}{2w_k^{\mathcal{V}}}(b_k^T\theta - \overline{P_{G_k}} + P_{L_k})^2, \quad \text{otherwise} \end{cases}$$

for constraint $b_k^T\theta - \overline{P_{G_k}} + P_{L_k} \leq 0$, $\forall k \in \mathcal{V}$ and

$$\underline{\chi_k}(\theta; w^{\mathcal{V}}) = \begin{cases} 0, \quad \text{if} \quad \underline{P_{G_k}} - P_{L_k} - b_k^T\theta \leq 0 \\ \frac{1}{2w_k^{\mathcal{V}}}(b_k^T\theta - \underline{P_{G_k}} + P_{L_k})^2, \quad \text{otherwise} \end{cases}$$

for constraint $b_k^T\theta - \underline{P_{G_k}} + P_{L_k} \geq 0$, $\forall k \in \mathcal{V}$.

Let us define the *the augmented cost* as

$$\begin{aligned} L(\theta; w^{\mathcal{E}}, w^{\mathcal{V}}) = \sum_{i\in\mathcal{V}} C_i((B\theta)_i + P_{L_i}) + \sum_{j\in\mathcal{E}} \overline{\psi_j}(\theta; w^{\mathcal{E}}) + \\ \sum_{j\in\mathcal{E}} \underline{\psi_j}(\theta; w^{\mathcal{E}}) + \sum_{i\in\mathcal{V}} \overline{\chi_i}(\theta; w^{\mathcal{V}}) + \sum_{i\in\mathcal{V}} \underline{\chi_i}(\theta; w^{\mathcal{V}}) \end{aligned} \tag{4.24}$$

$L(\theta; w^{\mathcal{E}}, w^{\mathcal{V}})$ is a continuous function of θ for fixed $w^{\mathcal{E}}$ and $w^{\mathcal{V}}$. Let $\theta^*[k]$ denote the approximate minimizer of $L(\theta; w^{\mathcal{E}}[k], w^{\mathcal{V}}[k])$. It was shown in [8] that by approximately minimizing $L(\theta; w^{\mathcal{E}}, w^{\mathcal{V}})$ for sequences of $\{w^{\mathcal{E}}[k]\}_{k=0}^{\infty}$ and $\{w^{\mathcal{V}}[k]\}_{k=0}^{\infty}$, the sequence of approximate minimizers $\{\theta^*[k]\}_{k=0}^{\infty}$ converges to the optimal point of the OPF problem. We only require that $\{w_j^{\mathcal{E}}[k]\}_{k=0}^{\infty}$ and $\{w_i^{\mathcal{V}}[k]\}_{k=0}^{\infty}$, $\forall j \in \mathcal{E}$, $\forall i \in \mathcal{V}$ are sequences of transmission line/generator penalty parameters that are monotone decreasing to zero.

Instead of minimizing $L(\theta; w^{\mathcal{E}}, w^{\mathcal{V}})$ for sequences of penalty parameters, we are only considering the problem of minimizing $L(\theta; w^{\mathcal{E}}, w^{\mathcal{V}})$ for fixed $w^{\mathcal{E}}$ and $w^{\mathcal{V}}$ in this chapter. If $w_j^{\mathcal{E}}$ and $w_i^{\mathcal{V}}$ are sufficiently small, the minimizer of $L(\theta; w^{\mathcal{E}}, w^{\mathcal{V}})$ will be a good approximation to the solution of the original OPF problem. We should note that in our work in [57], we gave an event-triggered algorithm that converges to the exact minimizer of the NUM problem. Interested reader can refer to that paper to see how we can decrease the penalty parameters in a distributed way to accomplish that.

We can search for the minimizer of $L(\theta; w^{\mathcal{E}}, w^{\mathcal{V}})$ using a gradient descent algorithm where

$$\theta_i(t) \quad = \quad -\int_0^t \nabla_{\theta_i} L(\theta(\tau); w^{\mathcal{E}}, w^{\mathcal{V}}) d\tau$$

for each generator $i \in \mathcal{V}$. The derivative of $L(\theta; w^{\mathcal{E}}, w^{\mathcal{V}})$ with respect to θ_i can be shown to be

$$\begin{aligned}
& \nabla_{\theta_i} L(\theta; w^{\mathcal{E}}, w^{\mathcal{V}}) \\
= & \sum_{j\in\mathcal{L}(i)} \max\{0, \frac{1}{w_j^{\mathcal{E}}}(a_j^T\theta - \overline{P_j})\}A_{ji} \\
+ & \sum_{j\in\mathcal{L}(i)} \min\{0, \frac{1}{w_j^{\mathcal{E}}}(a_j^T\theta - \underline{P_j})\}A_{ji} \\
+ & \sum_{k\in\mathcal{N}(i)+i} \max\{0, \frac{1}{w_k^{\mathcal{V}}}(b_k^T\theta - \overline{P_{G_k}} + P_{L_k})\}B_{ki} \\
+ & \sum_{k\in\mathcal{N}(i)+i} \min\{0, \frac{1}{w_k^{\mathcal{V}}}(b_k^T\theta - \underline{P_{G_k}} + P_{L_k})\}B_{ki} \\
+ & \sum_{k\in\mathcal{N}(i)+i} \nabla C_k(b_k^T\theta + P_{L_k})B_{ki}
\end{aligned}$$

For each transmission line, let us define

$$\mu_j(t) = \max\{0, \frac{1}{w_j^{\mathcal{E}}}(a_j^T\theta(t) - \overline{P_j})\} + \min\{0, \frac{1}{w_j^{\mathcal{E}}}(a_j^T\theta(t) - \underline{P_j})\} \quad (4.25)$$

Here $a_j^T\theta(t)$ is simply the power flow on transmission line $j \in \mathcal{E}$ at time t. $w_j^{\mathcal{E}} > 0$ is a constant penalty coefficients that penalizes the violation of the transmission line flow limits. It is easy to see that $\mu_j(t)$ is nonzero if and only if the flow on the jth transmission line exceeds the line flow limits. $\mu_j(t)$ summarizes the information of the jth transmission line at time t and can be viewed as its state.

Similarly for each generator $k \in \mathcal{V}$ we define

$$\begin{aligned}
\varphi_k(t) = & \nabla C_k(b_k^T\theta(t) + P_{L_k}) \\
+ & \max\{0, \frac{1}{w_k^{\mathcal{V}}}(b_k^T\theta(t) + P_{L_k} - \overline{P_{G_k}})\} + \min\{0, \frac{1}{w_k^{\mathcal{V}}}(b_k^T\theta(t) + P_{L_k} - \underline{P_{G_k}})\}
\end{aligned}$$

Here $w_k^{\mathcal{V}}$ is a constant penalty coefficient that penalizes the violation of the generating limits for generator $k \in \mathcal{V}$. Recall the power balance equation in 4.14 and we can thus rewrite $\varphi_k(t)$ as

$$\begin{aligned}\varphi_k(t) &= \nabla C_k(P_{G_k}(t)) + \max\{0, \frac{1}{w_k^{\mathcal{V}}}(P_{G_k}(t) - \overline{P_{G_k}})\} \\ &+ \min\{0, \frac{1}{w_k^{\mathcal{V}}}(P_{G_k}(t) - \underline{P_{G_k}})\}\end{aligned}$$

It is then easy to see that $\varphi_k(t)$ is determined by the gradient of the current generation cost of the kth generator, and whether the kth generator's generation limit is satisfied. In other words, $\varphi_k(t)$ summarizes the information of the kth generator at time t and can be viewed as its state.

We can now rewrite the derivative of $L(\theta; w^{\mathcal{E}}, w^{\mathcal{V}})$ with respect to θ_i as

$$\nabla_{\theta_i} L(\theta; w^{\mathcal{E}}, w^{\mathcal{V}}) = \sum_{j \in \mathcal{L}(i)} \mu_j A_{ji} + \sum_{k \in \mathcal{N}(i)+i} \varphi_k B_{ki} \tag{4.26}$$

and the gradient descent algorithm takes the form

$$\theta_i(t) = -\int_0^t \left[\sum_{j \in \mathcal{L}(i)} \mu_j(\tau) A_{ji} + \sum_{k \in \mathcal{N}(i)+i} \varphi_k(\tau) B_{ki} \right] d\tau \tag{4.27}$$

Note that in equation 4.27, generator i can compute its phase angle only based on the state φ_i from itself and its neighboring generators, as well as the state μ_j from its outgoing transmission lines. The update of θ_i can be done in a distributed manner.

However, in the above equation, this neighboring generator's state information is available to generator i in a continuous manner, which would require continuous communications between neighboring generators. This is highly undesirable. An *event-triggered* version of equation 4.27 assumes that generator i accesses a *sampled* version of its neighboring generator's state. In particular, let's associate a sequence of *sampling* instants, $\{T_i[\ell]\}_{\ell=0}^{\infty}$ with the ith generator. The time $T_i[\ell]$ denotes the instant when the ith generator samples its state φ_i for the ℓth time and transmits that state to neighboring generators $k \in \mathcal{N}(i)$. We can see that at any time $t \in \Re$, the sampled generator state is a piecewise constant function of time in which

$$\hat{\varphi}_i(t) = \varphi_i(T_i[\ell]) \tag{4.28}$$

for all $\ell = 0, \cdots, \infty$ and any $t \in [T_i[\ell], T_i[\ell+1])$. In this regard, the "event-triggered" version of equation 4.27 takes the form

$$\theta_i(t) \;\; = \;\; -\int_0^t \left[\sum_{j \in \mathcal{L}(i)} \mu_j(\tau) A_{ji} + \sum_{k \in \mathcal{N}(i)} \hat{\varphi}_k(\tau) B_{ki} + \varphi_i(\tau) B_{ii} \right] d\tau \tag{4.29}$$

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

The sequence $\{T_i[\ell]\}_{\ell=0}^{\infty}$ represents time instants when generator i transmits its "state" to its neighboring generators. Here we assume that there is no transmission delay in each $\hat{\varphi}_i(t)$ broadcast.

Next we will state the main theorem of this subsection, which states the condition under which each generator should sample and broadcast its state.

Theorem 4.4.1 *Consider the Lagrangian in equation 4.24 where the functions C_i are differentiable, strictly increasing, and convex. Assume fixed generator and transmission line penalty parameters $w^{\mathcal{E}} > 0$, $w^{\mathcal{V}} > 0$. $\forall i \in \mathcal{V}$, define*

$$z_i(t) = \sum_{j \in \mathcal{L}(i)} \mu_j(\tau) A_{ji} + \sum_{k \in \mathcal{N}(i)} \hat{\varphi}_k(\tau) B_{ki} + \varphi_i(\tau) B_{ii}$$

and

$$\rho_i = 1 / \sqrt{\sum_{k=1}^{N} |\mathcal{N}(k)| B_{ik}^2} \tag{4.30}$$

Consider the sequences $\{T_i[\ell]\}_{\ell=0}^{\infty}$ *for each* $i \in \mathcal{V}$. *For each generator* $i \in \mathcal{V}$, *let its phase angle,* $\theta_i(t)$, *satisfy equation 4.29 with sampled neighboring state given by equation 4.28. Assume that for all* $i \in \mathcal{V}$ *and all* $\ell = 0, \cdots, \infty$, *that*

$$|\varphi_i(t) - \hat{\varphi}_i(t)| \leq \rho_i |z_i(t)| \tag{4.31}$$

for $t \in [T_i[\ell], T_i[\ell+1])$. *Then the phase angle* $\theta(t)$ *asymptotically converge to the unique minimizer of* $L(\theta; w^{\mathcal{E}}, w^{\mathcal{V}})$. ■

Proof: For convenience, we do not explicitly include the time dependence in the proof.

For all $t \geq 0$ we have

$$\begin{aligned}
& -\dot{L}(\theta; w^{\mathcal{E}}, w^{\mathcal{V}}) \\
= \;& -\sum_{i=1}^{N} \frac{\partial L}{\partial \theta_i} \frac{d\theta_i}{dt} \\
= \;& \sum_{i=1}^{N} z_i \left[\sum_{j \in \mathcal{L}(i)} \mu_j A_{ji} + \sum_{k \in \mathcal{N}(i)} \varphi_k B_{ki} + \varphi_i B_{ii} \right] \\
\geq \;& \frac{1}{2} \sum_{i=1}^{N} \left\{ z_i^2 - \left[\sum_{k \in \mathcal{N}(i)} (\varphi_k - \hat{\varphi}_k) B_{ki} \right]^2 \right\} \\
\geq \;& \frac{1}{2} \sum_{i=1}^{N} z_i^2 - \frac{1}{2} \sum_{i=1}^{N} |\mathcal{N}(i)| \sum_{k \in \mathcal{N}(i)} \left[(\varphi_k - \hat{\varphi}_k) B_{ki} \right]^2 \\
= \;& \frac{1}{2} \sum_{i=1}^{N} z_i^2 - \frac{1}{2} \sum_{i=1}^{N} |\mathcal{N}(i)| \sum_{k=1}^{N} \left[(\varphi_k - \hat{\varphi}_k) B_{ki} \right]^2 \\
= \;& \frac{1}{2} \sum_{i=1}^{N} z_i^2 - \frac{1}{2} \sum_{k=1}^{N} (\varphi_k - \hat{\varphi}_k)^2 \sum_{i=1}^{N} |\mathcal{N}(i)| B_{ki}^2 \\
= \;& \frac{1}{2} \sum_{i=1}^{N} z_i^2 - \frac{1}{2} \sum_{i=1}^{N} \left(\sum_{k=1}^{N} |\mathcal{N}(k)| B_{ik}^2 \right) (\varphi_i - \hat{\varphi}_i)^2
\end{aligned}$$

which immediately suggests that if the sequences of sampling instants $\{T_i[\ell]\}_{\ell=0}^{\infty}$ satisfy the inequality in equation 4.31 for all $\ell = 0, 1, 2, ..., \infty$, and any $i \in \mathcal{V}$, then $\dot{L}(\theta; w^{\mathcal{E}}, w^{\mathcal{V}}) \leq 0$ is guaranteed for all t.

By using the properties of C_i and $\overline{\psi_j}(\theta; w^{\mathcal{E}})$, $\underline{\psi_j}(\theta; w^{\mathcal{E}})$, $\overline{\chi_k}(\theta; w^{\mathcal{V}})$, $\underline{\chi_k}(\theta; w^{\mathcal{V}})$, it is easy to show that for any fixed $w^{\mathcal{E}}$ and $w^{\mathcal{V}}$, $L(\theta; w^{\mathcal{E}}, w^{\mathcal{V}})$ is strictly convex in θ. It thus has a unique minimizer. Suppose $\theta^*(w^{\mathcal{E}}, w^{\mathcal{V}})$ is this minimizer, and the corresponding Lagrangian is $L(\theta^*; w^{\mathcal{E}}, w^{\mathcal{V}})$. Define $V(\theta) = L(\theta; w^{\mathcal{E}}, w^{\mathcal{V}}) - L(\theta^*; w^{\mathcal{E}}, w^{\mathcal{V}})$. It is trivial to see $V(\theta)$ is a Lyapunov function for the system. Moreover, $\dot{V}(\theta) = 0$ means $\dot{L}(\theta; w^{\mathcal{E}}, w^{\mathcal{V}}) = 0$. The only scenario this can happen is at the equilibrium. As a result, the equilibrium $\theta^*(w^{\mathcal{E}}, w^{\mathcal{V}})$ is asymptotically stable. Proof complete. ■

Theorem 4.4.1 basically states an event-triggered distributed algorithm. This theorem asserts that each generator i's phase angle $\theta_i(t)$ needs to follow the direction suggested by equation 4.29. When the inequality in equation 4.31 is violated, generator i will trigger the sampling and transmission of generator state φ_i to its neighboring generators. Generator i compares the error between the last transmitted state $\hat{\varphi}_i$ and the current state φ_i. At the sampling time $T_i[\ell]$, this difference is zero and the inequality is trivially satisfied. After that time, the difference increases and when the inequality in equation 4.31 is violated, we let that time be the next sampling instant, $T_i[\ell + 1]$ and then transmit the sampled generator state $\hat{\varphi}_i$ to the neighboring generators $k \in \mathcal{N}_i$. The theorem asserts that if all the generators behave in the above way, then the generated power of all generators will approach the solution of the OPF problem.

It turns out that the above algorithm can be easily integrated into the CERTS microgrid controller. It is achieved by dynamically adjusting the requested power P_{req} for each generator. In the microsource controller in [32], generator i's phase angle θ_i is adjusted by comparing the measured active power P_{G_i} and the requested power $P_{req,i}$, where θ_i follows

$$\dot{\theta}_i(t) = \pi(P_{req,i} - P_{G_i}(t)) \tag{4.32}$$

This suggests that if instead of fixing $P_{req,i}$, which is what has been done in [32], we can adjust $P_{req,i}$ so that the direction suggested by equation 4.32 matches the direction suggested by our event-triggered scheme in equation 4.29. This can be easily done by setting

$$P_{req,i}(t) = P_{G_i}(t) - \frac{\gamma z_i(t)}{\pi} \tag{4.33}$$

Here $\gamma > 0$ is a constant that controls how fast we adjust the phase angle. This constant is necessary because if $z_i(t)$ is large, the adjustment in θ_i may be too fast, which as a result may destabilize the system. Since generator i can compute both P_{G_i} and z_i locally, $P_{req,i}$ can be easily computed by generator i itself. This suggests that each generator only needs to adjust its power set point according to equation 4.33. It samples and then broadcasts its state φ_i to its neighboring generators when the inequality in equation 4.31 is violated. If every generator follows this action, then by theorem 4.4.1, the generated power P_G of all generators will approach the solution of OPF.

4.5 Simulation

This section presents simulation results. The simulation is done in MATLAB/SimPower and shows that our algorithm indeed solves the OPF problem in a distributed way, and the communication between neighboring subsystems is very infrequent.

We built a simulation model in MATLAB/SimPower for the UWM CERTS microgrid controller in [32]. The system model (which consists of three buses) is shown in figure 4.4, and the generator model is shown in figure 4.5, which mimics the droop characteristics in traditional synchronous generators. We use a three bus example, which is shown in figure 4.6. The network consists of three generators with power set point $0.4, 0.8, 0.6$ (pu) respectively. There are three active loads which request $0.96, 0.72, 0.48$ (pu) active power, respectively. Transmission lines

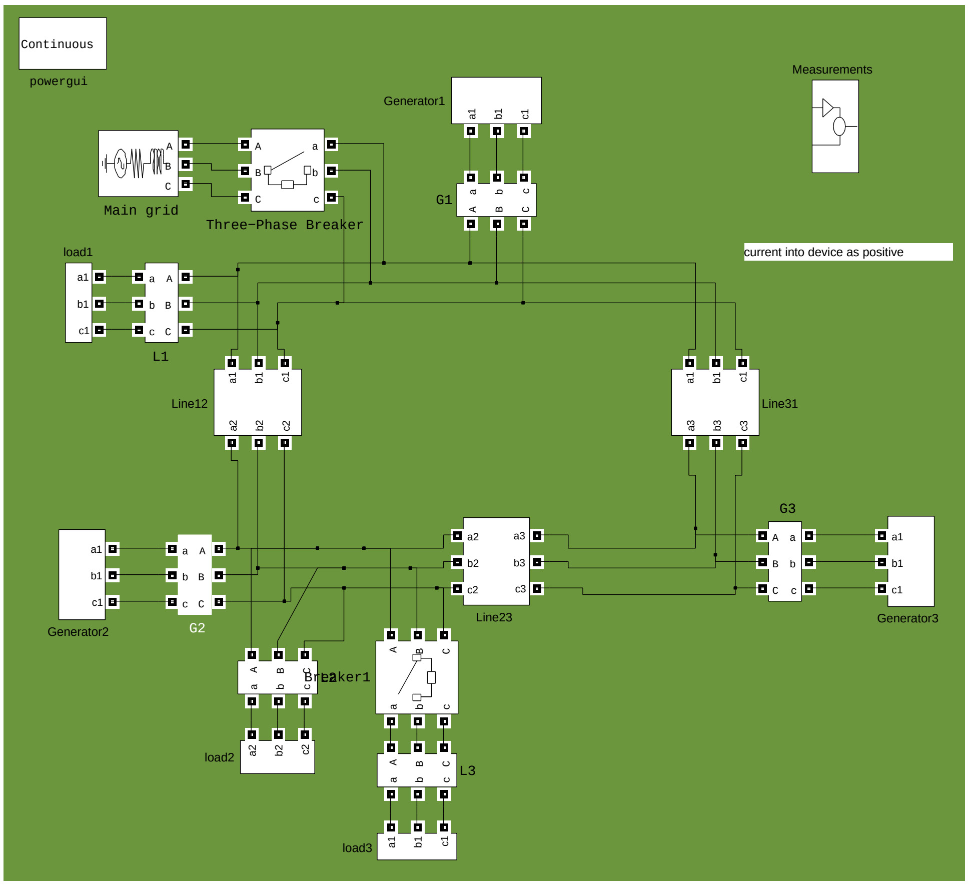

Figure 4.4. SimPower simulation model

are assumed to have zero resistance and all have impedances of $z = 0.0039j$. Each generator has generating limits between 0pu and 1pu. Each transmission line has power flow limits between -0.4pu to 0.4pu. The cost functions of the three generators are: $2.0 + 0.1p + 0.1p^2$, $3.0 + 1.8p + 0.1p^2$, $1.0 + 0.5p + 0.1p^2$. All generators come online at $t = 0$s with their initial fixed power set points. At $t = 3$s, we switch from the fixed power set point scheme to our event-triggered set point scheme. At $t = 10$s, the third load is added to bus 2.

Figure 4.7 plots the generator power as a function of time. The left three plots

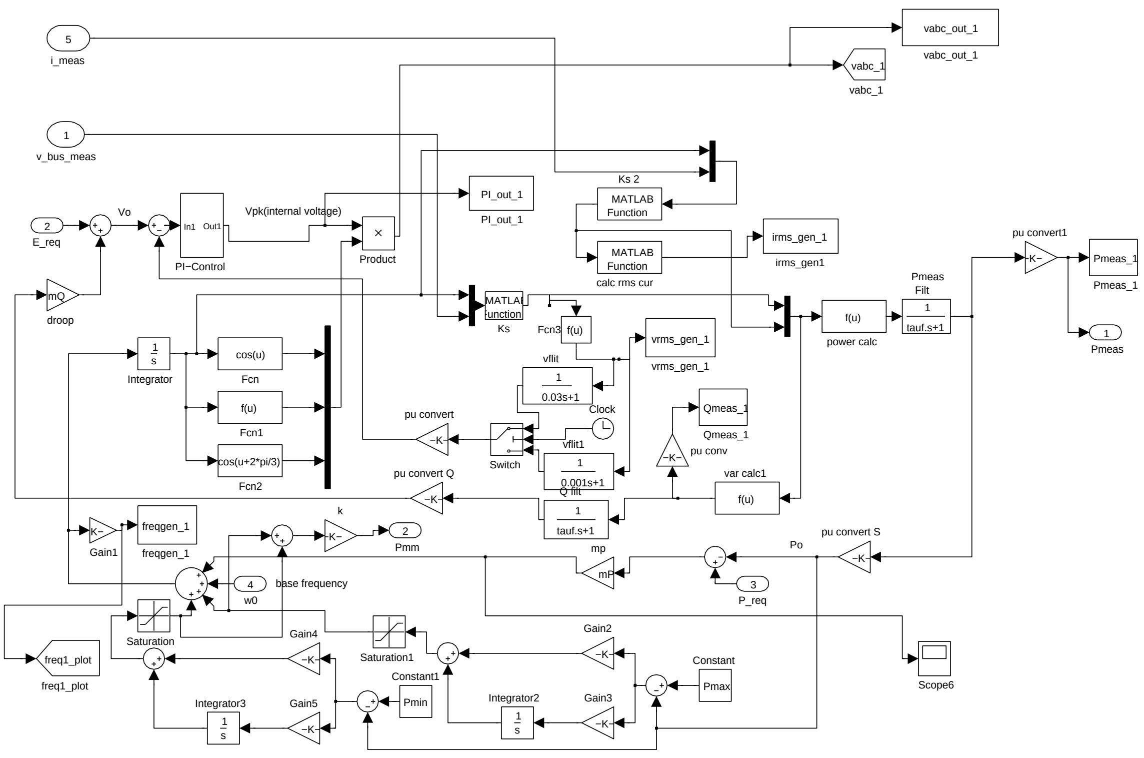

Figure 4.5. SimPower generator model

correspond to generators' measured power, and the right three plots correspond to the set points computed by equation 4.33 for three generators. We can see that the actual measured power tracks the computed set point very well. After switching to the event-triggered scheme at $t = 3$s, generator 1 quickly increases its generation to full capacity, because it has the lowest generation cost. At the same time, generator 2's generation drops to minimum, because it is the most expensive generator. When the new load is added at $t = 10$s, generator 1 cannot increase its generation further since it is already at full capacity, so generator 2 picks up the additional load. Figure 4.7 shows that our event-triggered scheme

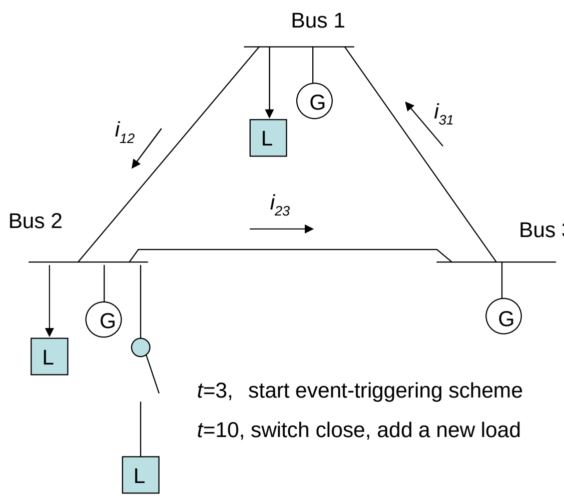
Bus 1
i_{12}
i_{31}
G
L
i_{23}
Bus 2
Bus 3
G
L
G
t=3, start event-triggering scheme
t=10, switch close, add a new load
L

Figure 4.6. Three generator simulation model

does adjust the power set point in a way that favors the low cost generator. Also the generating limit constraint is satisfied when using our new controller to adjust set point.

Figure 4.8 plots the instantaneous frequencies of the generators as a function of time for all three generators. We can easily notice that the frequencies are maintained at around 60Hz, which is highly desirable. This means the power generated by the generators are of high quality.

Figure 4.9 plots the power flow on the transmission line as a function of time for all three transmission lines. The important thing to notice here is that in the middle plot, the power flow on the transmission line is kept below 0.4pu, the flow limit of the line. At $t = 10$s, because of the addition of a load, the flow limit on the second transmission line is violated for less than 1s, but the power flow quickly adjusts back to within the limit. Both figure 4.7 and figure 4.9 show that our algorithm reacts very quickly to the changes in the network.

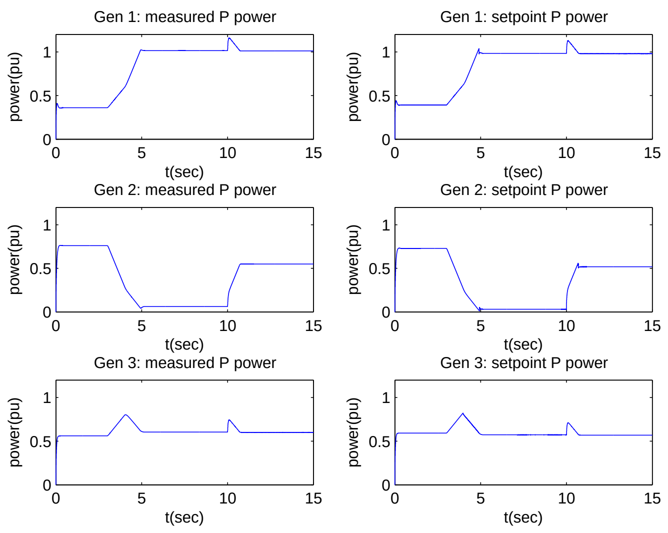

Figure 4.7. Measured generator power and generator power set point

Figure 4.10 plots the load power as a function of time for all three loads. All loads are well served. At $t = 10$s, the oscilliation brought by the addition of a load stabilizes in less than a second. The algorithm ensures that the load demand in the network is met and reacts quickly to the change in load demand.

Figure 4.11 plots the broadcast periods of the generators as a function of time. The y-axis represents the time passed since the last broadcast. This figure shows some very interesting result. Both the second and the third generator have broadcasted only twice, and the first generator broadcasted 9 times. If we compare figure 4.11 with figure 4.7, we will find out that only the first generator's generating limit constraint is active. This explains why it triggers much more often. For generator 2 and 3, they only need to broadcast their states occasionally or when some network condition changes. For generator 1, it has an active generating limit

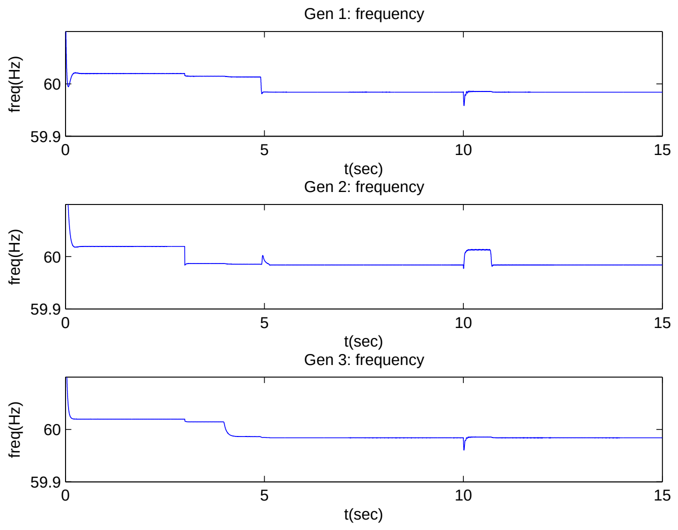

Figure 4.8. Generator frequencies

constraint, so it is more likely to broadcast its state. As we can see from the figure, most of the time the generators do not need to communicate with their neighbors at all (almost 5 secs for generator 2), this is highly desirable and has the potential to significantly reduce the communicate costs of a large scale power system.

Finally in figure 4.12 we plot the total generation cost as a function of time. Without any surprise, our scheme reduces the initial generation cost of 7.8 when using fixed set point scheme to about 6.6. This is a reduction of about 15%, which is significant.

4.6 Conclusion

This chapter uses the optimal power flow problem in microgrids as a nontrivial real-life example to demonstrate the effectiveness of event-triggered optimization

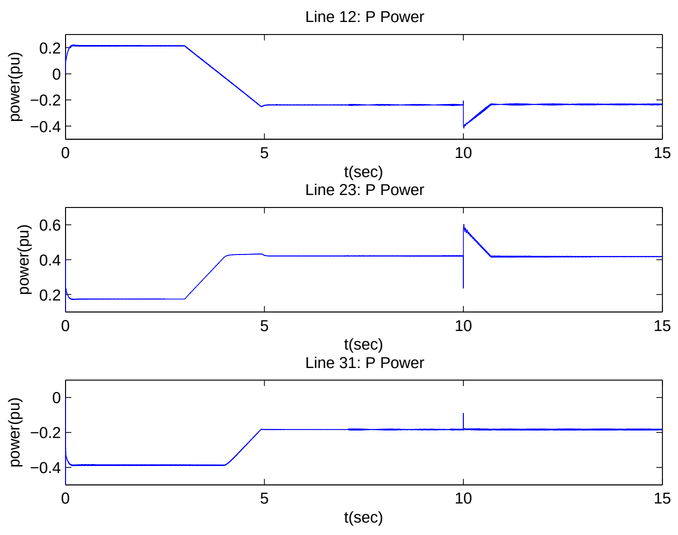

Figure 4.9. Transmission line power flow

algorithms. The chapter first formally states the OPF problem and introduces the UWM CERTS microgrid controller. It then presents an event-triggered distributed optimization algorithm for the OPF problem and proves its convergence. The MATLAB/SimPower simulation we bulit uses the CERTS microgrid model, and shows that the algorithm indeed solves the OPF problem while keeping communication between neighboring subsystems very infrequent.

There are several future directions we will pursue. First, the simulation considered in this chapter is a rather small example, and we would like to exploit more realistic larger scale simulations. Second, in our integration with the CERTS model, we simply 'replace' the existing controller in the model with our new event-triggered controller. Our new controller does guarantee the convergence to the solution of OPF, however, it does not provide guarantees of the transient sta-

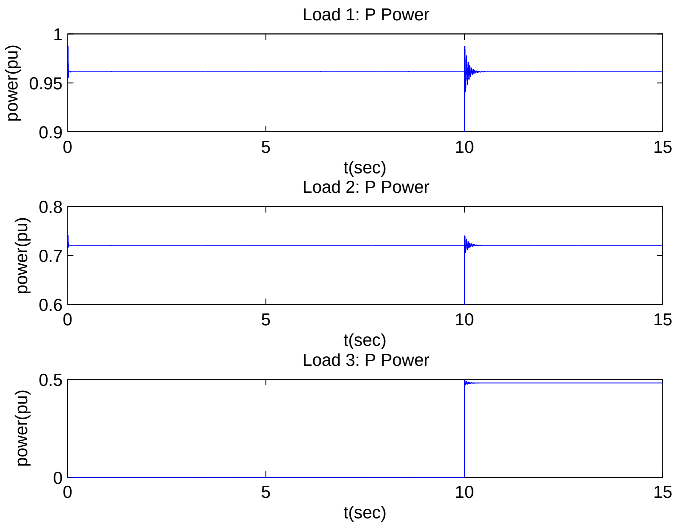

Figure 4.10. Load power

bility of the system. Future research will address this issue. Third, the algorithm we used in this chapter assumes that each generator adjusts the requested power input continuously, and also has continuous measurement of the power flowing away from each outgoing transmission lines. This however is not very realistic in most scenarios. Finally, we believe the load shedding problem can be viewed as a dual problem of the OPF problem, and we believe it can be solved using the same technique shown in this chapter.

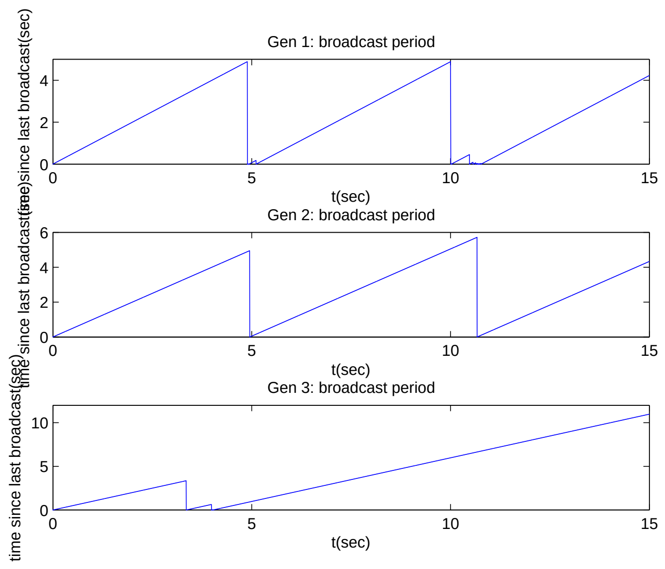

Figure 4.11. Broadcast period of generators

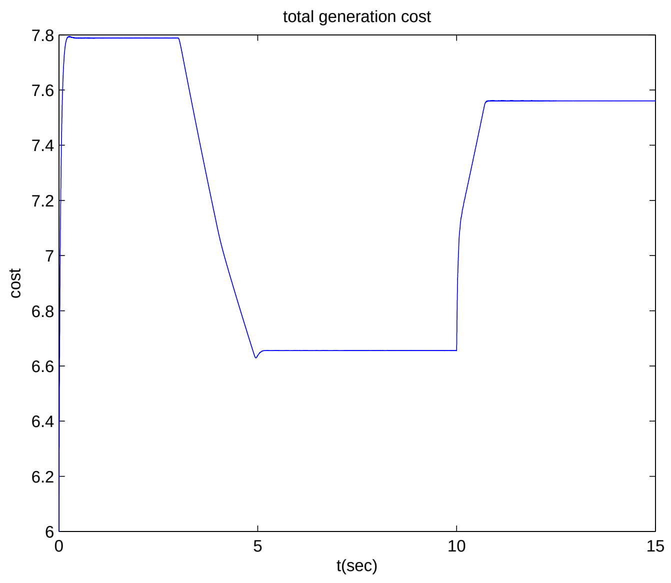

Figure 4.12. Total generation cost

CHAPTER 5

Conclusions

5.1 Summary of contributions

In this work, we are interested in reducing the message passing complexity of distributed optimization algorithms using event-triggering. The motivation comes from the fact that existing distributed optimization algorithms usually rely on some conservative choice of step size to ensure the convergence of the algorithm. The tight coupling between inter-subsystem communication and local computation brings in unnecessarily large message passing complexity in the algorithm. 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 [21]. 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.

In chapter 2 we start to study how to use event-triggering to separate communication and computation in distributed optimization algorithms. The chapter uses the NUM problem formulation as an example and presents a distributed algorithm based on barrier methods that uses event-triggered message passing. Under event triggering, each subsystem broadcasts to its neighbors when a local "error" signal exceeds a state dependent threshold. We prove that the proposed algorithm converges to the global optimal solution of the NUM problem. The proof is done in two steps. First, we prove the algorithm converges to the local minimizer when the barrier parameters are fixed. We then prove that when the barrier parameters are monotone decreasing, the sequence of approximate local minimizers indeed converge to the global minimizer, which is the 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 when compared to dual decomposition algorithms, and is scale-free with respect to two measures of network size $\overline{L}$ and $\overline{S}$. The work in this chapter appears in [58] [54].

The event-triggered barrier algorithm in chapter 2 is our very first approach to solve the distributed optimization problems using the event-triggered idea, and it shows significantly reduction in the communication cost of the network. However, as we show in the chapter, the algorithm suffers from issues like the need for an initial feasible point, and performance sensitive to parameter choices. All these issues limit the usefulness the algorithm. This inspired us to look for alternative algorithms to apply the event-triggered idea. We choose the augmented Lagrangian method instead of the barrier method in our following work.

Chapter 3 still uses the NUM problem formulation as an example and presents two distributed algorithms based on the augmented Lagrangian methods that use event-triggered message passing and proves their convergence. One of the algorithm is a primal algorithm, and the other is a primal-dual algorithm. We could present algorithms that converge to the exact minimizer instead of the approximate solution, as we have shown in [57]. However, we feel it unnecessary to present it here since the proof techniques are very similar to what we have shown in chapter 2. For this reason, both algorithms in this chapter only converge to an approximate minimizer to the NUM problem, which is enough for most applications.

For the primal-dual algorithm, we consider scenarios when the network has data dropouts, and give an upper bound on the largest number of successive data dropouts each link can have, while ensuring the asymptotic convergence of the algorithm. A state-dependent lower bound on the broadcast period and an upper bound on the transmission delay the network can tolerate while ensuring convergence are also given. Simulation results show that both algorithms have a message passing complexity that is up to two orders of magnitude lower than dual decomposition algorithms, and are scale-free with respect to two measures of network size $\overline{L}$ and $\overline{S}$. The primal algorithm in this chapter is similar to the algorithm in chapter 2. However, in chapter 2, 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, need for an initial feasible point and sensitive to the choice of parameters. The event-triggered algorithms presented in this chapter do not have these issues. Also, both algorithms have slightly better performance than the barrier method based algorithm in chapter 2 in terms of message passing complexity. The work in this chapter appears in [57] [59] [53] [55].

The problems and algorithms developed in the previous chapters are somewhat abstract, and did not explicitly show how we can use those event-triggered algorithms in real-life applications. Chapter 4 then uses the optimal power flow problem in microgrids as a nontrivial real-life example to demonstrate the effectiveness of event-triggered optimization algorithms.

The optimal power flow (OPF) problem has been the research subject of power system community since early 1960's, and is very similar to the NUM problem we considered in previous chapters. Various centralized or distributed optimization algorithms have been proposed to solve the OPF problem. These algorithms usually made the assumptions that communication between subsystems was not expensive and reliable, which is unrealistic. One way around this problem is to make use of low power ad hoc wireless communication networks that operate independently of the main power grid. Using event-triggering on those wireless networks therefore may provide a useful approach for reducing the power grid's use of the communication network.

In chapter 4, we develop an event-triggered distributed optimization algorithm for the OPF problem and prove its convergence. We use the CERTS microgrid model [32] as an example power system to show the effectiveness of our algorithm. The simulation is done in MATLAB/SimPower and shows that our algorithm solves the OPF problem in a distributed way, and the communication between neighboring subsystems is very infrequent. The work in this chapter appears in [56].

To our best knowledge, our work is the first examination of using the event-triggered idea to separate communication and computation in distributed optimization algorithms.

5.2 Future work

In this section, we discuss several possible future research directions for the work.

For the work in chapter 2 and 3, it would be useful quantify what penalty parameters w we need to choose in order to converge to a given desirable neighborhood of the optimal point. It would also be interesting to see the complexity and convergence time of both event-triggered algorithms analytically.

For the work in chapter 4, there are also several possible future directions we will pursue. First, the simulation considered in this chapter is a rather small example, and we would like to exploit more realistic larger scale simulations. Second, in our integration with the CERTS model, we simply 'replace' the existing controller in the model with our new event-triggered controller. Our new controller does guarantee the convergence to the solution of OPF, however, it does not provide guarantees of the transient stability of the system. Future research will address this issue. Third, the algorithm we used in this chapter assumes that each generator adjusts the requested power input continuously, and also has continuous measurement of the power flowing away from each outgoing transmission lines. This however is not very realistic in most scenarios. Finally, we believe the load shedding problem can be viewed as a dual problem of the OPF problem, and we believe it can be solved using the same technique shown in this chapter.

BIBLIOGRAPHY

1. M. Adler, J.Y. Cai, JK Shapiro, and D. Towsley. Estimation of congestion price using probabilistic packet marking. In *IEEE INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3.

2. E. Altman, T. Basar, R. Srikant, and S.A. INRIA. Robust rate control for ABR sources. *INFOCOM'98. Proceedings. IEEE*, 1.

3. K.E. Årzén. A simple event based PID controller. In *Proc. 14th IFAC World Congress*, 1999.

4. K.J. Astrom and B.M. Bernhardsson. Comparison of riemann and lebesgue sampling for first order stochastic systems. In *Proceedings of the IEEE Conference on Decision and Control*, volume 2, pages 2011–2016, 2002.

5. S. Athuraliya and S.H. Low. Optimization Flow Control, II: Implementation.

6. S. Athuraliya, SH Low, VH Li, and Q. Yin. REM: Active queue management. *IEEE network*, 15(3):48–53, 2001.

7. AR Bergen. Power Systems Analysis. *Englewood Cliffs, NJ.*

8. D.P. Bertsekas. *Nonlinear programming.* Athena Scientific, 1999.

9. PN Biskas, AG Bakirtzis, NI Macheras, and NK Pasialis. A decentralized implementation of DC optimal power flow on a network of computers. *IEEE Transactions on Power Systems*, 20(1):25–33, 2005.

10. B. Bollobás. *Modern Graph Theory.* Springer, 1998.

11. MF Carvalho, S. Soares, and T. Ohishi. Optimal active power dispatch by network flow approach. *IEEE Transactions on Power Systems*, 3(4):1640–1647, 1988.

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

13. W.P. Chen and L. Sha. An energy-aware data-centric generic utility based approach in wireless sensor networks. *IPSN*, pages 215–224, 2004.

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

15. M. Chiang, S.H. Low, A.R. Calderbank, and J.C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, 2007.

16. RD Christie, BF Wollenberg, and I. Wangensteen. Transmission management in the deregulated environment. *Proceedings of the IEEE*, 88(2):170–195, 2000.

17. AJ Conejo and JA Aguado. Multi-area coordinated decentralized DC optimal power flow. *IEEE Transactions on Power Systems*, 13(4):1272–1278, 1998.

18. AL Dimeas and ND Hatziargyriou. Operation of a Multiagent System for Microgrid Control. *Power Systems, IEEE Transactions on*, 20(3):1447–1455, 2005.

19. S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on networking*, 1(4):397–413, 1993.

20. S. Granville, E. CEPEL, P.R. Center, and R. de Janeiro. Optimal reactive dispatch through interior point methods. *Power Systems, IEEE Transactions on*, 9(1):136–146, 1994.

21. W.R. 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*, volume 2, 2000.

22. CA Hernandez-Aramburo, TC Green, N. Mugniot, and I.C. London. Fuel consumption minimization of a microgrid. *Industry Applications, IEEE Transactions on*, 41(3):673–681, 2005.

23. YC Ho, L. Servi, and R. Suri. A CLASS OF CENTER-FREE RESOURCE ALLOCATION ALGORITHMS'. In *Large Scale Systems Theory and Applications: Proceedings of the IFAC Symposium, Toulouse, France, 24-26 June 1980*, page 475. Franklin Book Co, 1981.

24. CV Hollot and Y. Chait. Nonlinear stability analysis for a class of TCP/AQM networks. *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, 3, 2001.

25. B. Johansson, M. Rabi, and M. Johansson. A simple peer-to-peer algorithm for distributed optimization in sensor networks. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 4705–4710, 2007.

26. B. Johansson, P. Soldati, and M. Johansson. Mathematical Decomposition Techniques for Distributed Cross-Layer Optimization of Data Networks. *Selected Areas in Communications, IEEE Journal on*, 24(8):1535–1547, 2006.

27. F.P. Kelly, A.K. Maulloo, and D.K.H. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49(3):237–252, 1998.

28. H.K. Khalil. *Nonlinear systems.* Prentice Hall Upper Saddle River, NJ, 2002.

29. BH Kim and R. Baldick. A comparison of distributed optimal power flow algorithms. *IEEE Transactions on Power Systems*, 15(2):599–604, 2000.

30. P. Kundur. *Power System Stability and Control.* McGraw-Hill Professional, 1994.

31. S. Kunniyur and R. Srikant. A time scale decomposition approach to adaptive ECN marking. *INFOCOM 2001. Proceedings. IEEE*, 3, 2001.

32. R. Lasseter. Control and design of microgrid components. Final Project Report - Power Systems Engineering Research Center (PSERC-06-03), 2006.

33. R.H. Lasseter and P. Piagi. Microgrid: A Conceptual Solution. *Proc. of 35th Annual IEEE Power Electronics Specialists Conference PESC04*, 6:4285–4290.

34. S.H. Low and D.E. Lapsley. Optimization flow control, I: basic algorithm and convergence. *IEEE/ACM Transactions on Networking (TON)*, 7(6):861–874, 1999.

35. R. Madan and S. Lall. Distributed algorithms for maximum lifetime routing in wireless sensor networks. In *IEEE GLOBECOM'04*, volume 2.

36. M. Mehyar, D. Spanos, and SH Low. Optimization flow control with estimation error. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 2, 2004.

37. L. Montestruque and M. Lemmon. Csonet: a metropolitan scale wireless sensor-actuator network. *Proceedings of the International Workshop on Mobile Device and Urban Sensing (MODUS)*, 2008.

38. J. Nocedal and S.J. Wright. *Numerical Optimization.* Springer, 1999.

39. R. Olfati-Saber, JA Fax, and RM Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

40. ARL Oliveira, S. Soares, and L. Nepomuceno. Optimal active power dispatch combining network flow and interior point approaches. *Power Systems, IEEE Transactions on*, 18(4):1235–1240, 2003.

41. F. Paganini. A global stability result in network flow control. *Systems and Control Letters*, 46(3):165–172, 2002.

42. DP Palomar and M. Chiang. Alternative Distributed Algorithms for Network Utility Maximization: Framework and Applications. *IEEE Transactions on Automatic Control*, 52(12):2254–2269, 2007.

43. Y. Qiu and P. Marbach. Bandwidth allocation in ad hoc networks: A price-based approach. In *Proceedings of IEEE INFOCOM 2003*, volume 2, pages 797–807.

44. M. Rabbat and R. Nowak. Distributed optimization in sensor networks. *Proceedings of the third international symposium on Information processing in sensor networks*, pages 20–27, 2004.

45. T. Ruggaber, J. Talley, and L. Montestruque. Using embedded sensor networks to monitor, control, and reduce CSO events: A pilot study. *Environmental Engineering Science*, 24(2):172–182, 2007.

46. J. Sandee, W. Heemels, and P. van den Bosch. Case studies in event-driven control. In *Hybrid Systems: Computation and Control*, pages 762–765. Springer, 2007.

47. A. Speranzon, C. Fischione, and K.H. Johansson. Distributed and Collaborative Estimation over Wireless Sensor Networks. *Proceedings of the IEEE Conference on Decision and Control*, pages 1025–1030, 2006.

48. P. Tabuada. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic Control*, 52(9):1680–1685, 2007.

49. J. Tsitsiklis, D. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, 1986.

50. Y.Z. Tsypkin. *Relay Control Systems.* Cambridge University Press, 1984.

51. R. Walter. *Principles of mathematical analysis.* McGraw-Hill, 1976.

52. P. Wan and M. Lemmon. Distributed Flow Control using Embedded Sensor-Actuator Networks for the Reduction of Combined Sewer Overflow (CSO) Events. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 1529–1534, 2007.

53. P. Wan and M. Lemmon. An event-triggered distributed primal-dual algorithm for Network Utility Maximization. In *Proceedings of the 48th IEEE Conference on Decision and Control*, 2009.

54. P. Wan and M. Lemmon. Distributed network optimization using event-triggerd barrier methods. *Submitted to European Journal of Control*, Aug 2009.

55. P. Wan and M. Lemmon. Distributed network optimization using event-triggered algorithms. *Submitted to IEEE Transactions on Automatic Control*, Mar 2009.

56. P. Wan and M. Lemmon. Optimal power flow in microgrid using event-triggered optimization algorithm. In *Submitted to 2010 American Control Conference*, Sep 2009.

57. P. Wan and M. D. Lemmon. Distributed Network Utility Maximization using Event-triggered augmented Lagrangian methods. In *Proceedings of American Control Conference*, 2009.

58. P. Wan and M. D. Lemmon. Distributed Network Utility Maximization using Event-triggered Barrier Methods. In *Proceedings of European Control Conference*, 2009.

59. P. Wan and M.D. Lemmon. Event-triggered distributed optimization in sensor networks. In *Proceedings of the 2009 ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 49–60, 2009.

60. X. Wang and M. Lemmon. Self-triggered feedback control systems with finite-gain L 2 stability. *IEEE Transactions on Automatic Control*, 54:452–467, 2009.

61. X. Wang and M.D. Lemmon. State based Self-triggered Feedback Control Systems with L2 Stability. In *Proceedings of the 17 IFAC World Congress*, 2008.

62. X. Wang and M.D. Lemmon. Event-triggering in distributed networked systems with data dropouts and delays. In *Proceedings of Hybrid Systems: computation and control*, 2009.

63. JT Wen and M. Arcak. A unifying passivity framework for network flow control. *IEEE Transactions on Automatic Control*, 49(2):162–174, 2004.

64. L. Xiao, M. Johansson, and SP Boyd. Simultaneous routing and resource allocation via dual decomposition. *IEEE Transactions on Communications*, 52(7):1136–1144, 2004.

65. Y. Xue, B. Li, and K. Nahrstedt. Optimal resource allocation in wireless ad hoc networks: a price-based approach. *IEEE Transactions on Mobile Computing*, 5(4):347–364, 2006.

66. Y. Yi and S. Shakkottai. Hop-by-hop congestion control over a wireless multi-hop network. *IEEE/ACM Transactions on Networking*, 15(1):133–144, 2007.

67. A. Zymnis, N. Trichakis, S. Boyd, and D. O'Neill. An interior-point method for large scale network utility maximization.