8-2016

Topology Design and Delay Control for Communication Networks in Smart Grid

Xiaodong Wang
University of Tennessee, Knoxville, xwang33@vols.utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Xiaodong Wang entitled "Topology Design and Delay Control for Communication Networks in Smart Grid." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Engineering.

Husheng Li, Major Professor

We have read this dissertation and recommend its acceptance:

Hairong Qi, Xueping Li, Chao Tian

Accepted for the Council:
Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Topology Design and Delay Control for Communication Networks in Smart Grid

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Xiaodong Wang

August 2016

Acknowledgements

First and foremost, I would like to thank my current advisor Dr. Husheng Li for the motivation and guidance on my dissertation. I would also like to thank Dr. Xiaorui Wang for guiding my research. Moreover, I would like to thank Dr. Harirong Qi, Dr. Chao Tian and Dr. Xueping Li for serving on my committee.

Abstract

Stability is a critical concern in the design and maintenance of power systems. Different approaches have been proposed for the analysis of power grid stability in various scenarios depending on small or large perturbations and the speed of the phenomenon of interest. In this work, we consider the power grid as a group of flocking birds, as synchronization is the key issue in both contexts. The framework of partial difference equation (PdE) is used to analyze the system stability, when designing the communication network of the power grid network for conveying measurements between different power stations. Both the cases where communication network delay is negligible and non-negligible are studied here. The communication network design problem is formulated as an optimization problem under the consideration of a stable power grid. Corresponding optimization algorithms are designed to solve the problem.

To convey measurements of the power network, wireless sensor networks is adopted, for its non-invasive and easy deployment properties. Periodic sleep scheduling is adopted to effectively save energy for the wireless sensor networks. To provide a controllable end-to-end delay for the communication networks, a dynamic duty cycle control approach is designed, featuring a single-hop delay controller based on the well known feedback control theory. The delay control approach also features a queuing delay adaptation scheme that adapts the duty cycle of each node to unpredictable packet rates, as well as a novel energy balancing approach that extends the network lifetime by dynamically adjusting the delay requirement allocated to each hop.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Power Grid Stability

The stability is a critical concern in the design and maintenance of power systems Kundur (1994). Various approaches have been proposed for the analysis of power grid stability in various scenarios depending on small or large perturbations and the speed of the phenomenon of interest. Different devices and algorithms work to stabilize the power grid following disturbances, such as Power System Stabilizers (PSSs) and flexible alternating current transmission systems (FACTS) devices. In the near future smart grids, phasor measurement units (PMUs) Phadke and Thorp (2008) will be widely deployed to collect the realtime synchronized information that will be critical for improved control of the power grid.

1.2 Power Grid and Flocking Birds

Flocking is used to describe a phenomenon where different individuals are moving in an ordered motion based only on very limited surrounding information and simple rules. For example, a large group of birds can fly in a approximate same direction with approximate same speed. In such a group, individual birds seems to move together as one. This phenomena has been studied extensively in biological science field, to

answer the questions as why the birds fly in a flock and how do they maintain the order of a flock. It turns out each bird atomically propel itself based on the distance from the birds in the vicinity and direction that other birds on flying into Quera et al. (2010).

Similarly the power grid in this study consists of by synchronized generators that cannot deviate far or for long from the nominal frequency. The dynamics of this interactions is similar to those of flocking birds, since in both cases each node needs to keep the relative 'distance' (namely, Euclidean distance in flocking birds and phase difference in generators) to others. As is well known, the system dynamics in power grids are characterized by the evolutions of phases and frequencies of different generators. We can consider each generator as a flying bird. The phase of each generator is analogous to the location of each bird in the flocking system, while the frequency of each generator is analogous to the speed of the corresponding bird. The difference of phase between generators is analogous to the distance between different birds. In the bird flocking system, the speed of each bird can be adjusted according to the distances to neighboring birds. Similarly, the frequency of each generator is also affected by the phase differences to neighboring generators. Moreover, the frequency can also be adjusted by the frequency measurements of neighboring generators. Since there have been substantial mathematical studies on the bird flocking, it is natural to introduce the research tools on flocking into the study of power grid stability.

1.3 PdE for network dynamics

In the seminal work Thorp et al. (1998), J. S. Thorp modeled the power grid as a medium continuous in both time and space, and thus described the electromechanical perturbations propagated in the power grid using a partial differential equation (PDE). Although the PDE modeling of power grid dynamics can facilitate the application of many existing mathematical tools in PDE, the assumption of a spatially continuous power grid is obviously not true, although it facilitates the analysis and

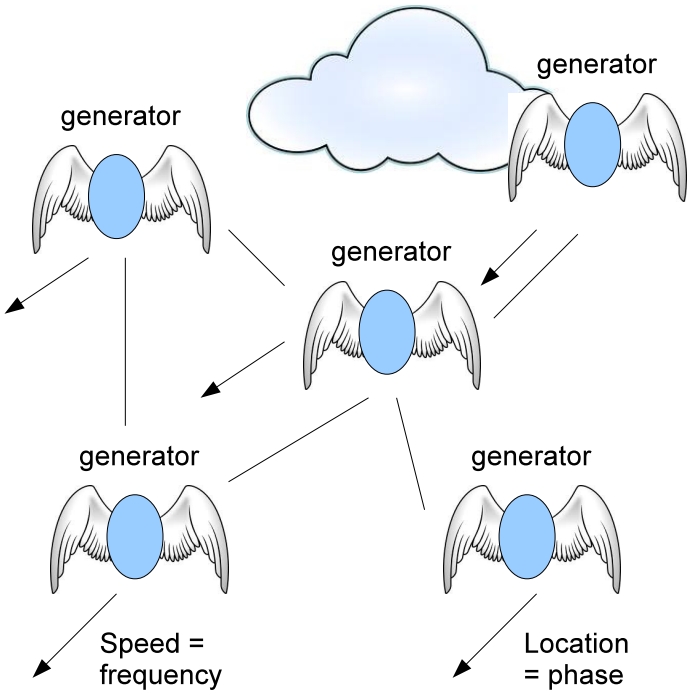

Figure 1.1: An illustration of flocking birds and generators.

provides insights; moreover, this assumption misses the information of power network topology and thus cannot evaluate the impact of the power network topology on the system dynamics. When incorporating the discrete power network topology, the space becomes discrete and thus the dynamics are determined by 'difference', instead of 'differential', in the space. Therefore, it is natural to employ the tool of PdE, which describes the dynamics in discrete networks and has been applied in the study of flocking Ferrari-Trecate et al. (2014).

1.4 Delay Control in Wireless Sensor Networks for Smart Grid

The current power grid is facing several challenges including increasing electricity demand, power distribution network congestion, and lack of pervasive and effective communications, etc. According to Gungor et al. (2010), several different blackouts in the past few years were caused by power network congestion and safety-relate factors. To address these challenges, the next generation power grid is being designed as a smart power grid. People can anticipate improved efficiency, safety and reliability in the smart power grid, using smart control and advanced communication systems and technology. In the smart grid, it is essentially important to build a resilient and online communication network for reliable information sharing between different power stations. With a reliable and online communication network, the impact of factors such as equipment failure and capacity limitation can be largely avoided. Gungor et al. (2010)

Traditional power system monitoring and diagnostics are typically conducted by wired communications. The problem with the wired communication infrastructure is the cost to deploy the communication cables are expensive and thus is are not widely implemented between power stations. To alleviate the cost problem and achieve reliable communications in the power grid, wireless sensor network is an ideal type

of infrastructure that can be used in the communication network in power grid to convey the measurement between different power station. There have been quite a few studies focusing on WSN application in power grid such as Tuna et al. (2013), Liu (2012) and Zhang et al. (2012).

To effectively use wireless sensor network in this scenario, two problems need to be solved. First, the end-to-end communication delay needs to be controllable. Second, the life time of the sensor network needs to last for a desired time. In this work, we propose DutyCon, a dynamic duty cycle control scheme that provides an end-to-end communication delay guarantee while taking advantage of periodic sleeping to achieve energy efficiency. DutyCon is significantly different from the existing work on delay and duty cycle management in WSNs in three aspects. First, we control the end-to-end delay of each data flow in a WSN to a user-specific bound while achieving energy conservation. Second, we dynamically adjust the duty cycle of each node individually to adapt to the network condition changes in different areas in the WSN. Network conditions in a WSN can vary both spatially and temporally Cerpa et al. (2005). Third, DutyCon can also adapt to the unpredictable incoming packet rate changes, which are common in many WSN-based monitoring applications, e.g., packets can be generated at a higher rate when an emergency event occurs.

Chapter 2

Stability in Power Grid with PdE

We first study the power grid stability based on the frameworks of flocking and PdE, where communication delay can be both negligible or not. In particular, we derive the conditions of system stability for both cases, based on which we propose an algorithm for designing the topology of communication network.

2.1 System Model

In this section, we introduce the system model. We first explain the dynamics of individual generators in power grids. Then, the interconnection among generators is briefed and we approximate the nonlinear dynamics using a linear one. Finally, we introduce the model of communication network in the smart grid.

2.1.1 Individual Generator

We consider a power network with N generators. For generator n, its dynamics is described by the following Swing Equation Kundur (1994):

$$M_n\ddot{\delta}(t) + D_n\dot{\delta}(t) = P_m^n(t) - P_e^n(t), \tag{2.1}$$

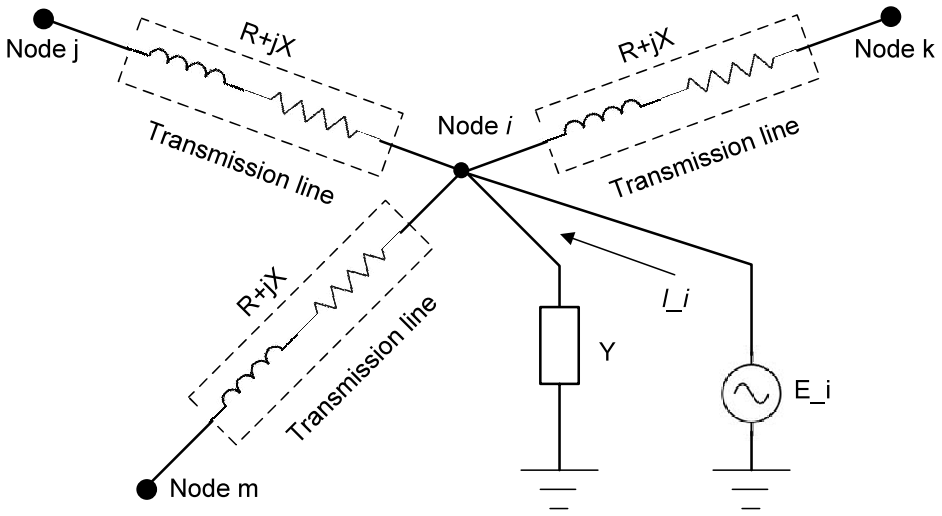

Figure 2.1: An illustration of one node in the power grid.

where δ is the phase, P_m^n is the mechanical power and P_e^n is the electric power. M_n is the rotor inertia constant and D_n is the mechanical damping constant. We denote by $f = \dot{\delta}$, whose physical meaning is the frequency of rotation. Then, the Swing Equation can be rewritten as

$$\begin{cases} \dot{\delta}(t) = f \\ M_n \dot{f}(t) + D_n f(t) = P_m^n(t) - P_e^n(t) \end{cases} . \tag{2.2}$$

2.1.2 Generator Interconnection

We consider the interconnections of the generators. Similarly to the seminal work by J. S. Thorp Thorp et al. (1998), we do not consider the connection of loads. It is non-trivial to incorporate the impact of electric loads, which will be our future study. However, the study on only generators can provide significant insights and pave the way to more general power network models.

Consider an arbitrary node i as shown in Fig. 2.1, where $Z_{ik} = R_{ik} + jX_{ik}$ is the impedance of the transmission line between the adjacent generators i and k, E_i is the voltage of generator i and Y_i is the shunt admittance. Then, for generator k adjacent

to i, which is denoted by $k \sim i$, the current in transmission line ij is given by

$$I_{i,k} = \frac{E_i - E_k}{Z_{ik}}. \tag{2.3}$$

Then, the current flowing from the voltage source to the node, denoted by I_i, is given by

$$\begin{aligned} I_i &= \sum_{k\sim i} I_{i,k} + Y_i E_i \\ &= \sum_{k\sim i} \frac{(E_i - E_k)}{Z_{ij}} + Y_i E_i. \end{aligned} \tag{2.4}$$

We assume that a perfect voltage control is applied such that the voltage of the generator is constant in magnitude. Hence, we have $E_i = Ve^{j\delta_i}$, where δ_i is the phase and V is the constant magnitude. Hence, the real power spent by generator i is given by

$$\begin{aligned} P_e^i &= Re\left[E_i I_i^*\right] \\ &= \sum\sum_{k\sim i} \frac{V^2 R_{ik}}{|Z_{ik}|^2} - \sum_{k\sim i} \frac{V^2 R_{ik}\cos(\delta_i - \delta_k)}{|Z_{ik}|^2} \\ &- \sum_{k\sim i} \frac{V^2 X_{ik}\sin(\delta_i - \delta_k)}{|Z_{ik}|^2} + V^2 Re[Y_i], \end{aligned} \tag{2.5}$$

which is obtained by substituting (2.4) into the expression of P_e^i. Substituting (2.5) into (2.2), we obtain the dynamics of the generators.

2.1.3 Linearization

Notice that Eq. (2.2) is nonlinear due to the nonlinearity of P_e^i in terms of $\delta_i - \delta_k$. It is very challenging to discuss the nonlinear dynamics directly. To simplify the analysis, we assume that the system is close to an equilibrium point. We denote the standard frequency by f_0 and the frequency deviation of generator i by Δf_i. The

angle deviation $\delta_i - f_0 t - \theta_i$ (θ_i is the initial phase of generator i) is denoted by $\Delta\delta_i$. Then, when Δf_i and $\Delta\delta_i$, $i = 1, ..., N$, are both sufficiently small, it is easy to verify that the dynamics can be linearized to the following form:

$$\begin{cases} \dot{\Delta\delta_i}(t) = \Delta f_i(t) \\ M_i \dot{\Delta f_i}(t) + D_i \Delta f_i(t) \\ = \Delta P_m^i(t) - \sum_{k\sim i} c_{ik} \left(\Delta\delta_i - \Delta\delta_k\right) \end{cases}, \tag{2.6}$$

where ΔP_m^i is the difference between the actual mechanical power and the stable one at the equilibrium, and

$$c_{ik} = \frac{V^2 R_{ik}}{|Z_{ik}|^2} \sin\delta_{ik} - \frac{V^2 X_{ik}}{|Z_{ik}|^2} \cos\delta_{ik}, \tag{2.7}$$

where δ_{ik} is the stable phase difference between adjacent generators i and k at the equilibrium state.

To facilitate the framework of PdE, we make the following assumptions which further simplifies the dynamics of the power grid:

- The damping constants are identical for all generators, which is denoted by D;
- The phase difference δ_{ik} is relatively small such that $\sin\delta_{ik} \approx 0$. Hence, we can assume

  $$c_{ik} = -\frac{V^2 X_{ik}}{|Z_{ik}|^2}, \tag{2.8}$$

  which implies $c_{ik} = c_{ki}$ since $\cos\delta_{ik} = \cos\delta_{ki}$.
- All the rotor inertias are all equal to 1, i.e., $M_n = 1$. This is for simplicity of notation and numerical simulations. It is easy to extend to the general case, since we do not use this assumption in the mathematical derivation.

For notational simplicity, we define $c_{ik} = 0$ if generators i and k are not adjacent.

2.1.4 Communications

We assume that there exist communications between some physically adjacent generators. We denote by $i \leftrightarrow j$ that generators i and j can communicate with each other*. A necessary condition for $i \leftrightarrow j$ is $i \sim j$. We assume that the mechanical power is adjusted according to the feedbacks of frequencies from the communication channels, i.e.,

$$\Delta P_m^i(t) = g\left(\Delta f_i(t), \{\Delta f_k(t-d)\}_{k \leftrightarrow i}\right), \tag{2.9}$$

where d is the communication delay of the communication link, which is assumed to be a constant†, and g is control policy of the mechanical power. Note that the change of the mechanical power can be realized through the governor, fast valving, the voltage control of the PSS or other devices Cvtkovic and Ilich (2011). Here, we do not explicitly model the detailed dynamics of such approaches.

For simplicity, we assume that the control of the mechanical power is a linear function, i.e.,

$$\Delta P_m^i(t) = \sum_{k \leftrightarrow i} g_{ik}\left(\Delta f_k(t-d) - \Delta f_i(t-d)\right). \tag{2.10}$$

We assume $g_{ik} = g_{ki}$; i.e., the control gains are symmetric. Similarly, we define $g_{ik} = 0$ if generators i and k cannot communicate with each other. We also assume that $g_{ik} \geq 0$ since it is desirable to decrease the mechanical power when Δf_i is large when compared with the frequency deviations of neighboring ndoes.

Incorporating the control action on the mechanical power into the overall dynamics, the linearized system dynamics can be written in the following vector

*Theoretically, if the communication network is connected, every two nodes can communicate with each other, via one or more hops. In this paper, we consider only a single hop communication.

†In practice, the communication delay could be different for different links, or even could be random, due to random channel conditions or communication congestions. The assumption of constant delay can substantially simplify the analysis and provide insights for our future study on the generic case of communication delay.

form:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{x}(t-d), \tag{2.11}$$

where $\mathbf{x} = (\Delta\delta_1, ..., \Delta\delta_N, \Delta f_1, ..., \Delta f_N)^T$. The matrices $\mathbf{A}$ and $\mathbf{B}$ are given by

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{L}_p & -D\mathbf{I} \end{pmatrix}, \tag{2.12}$$

where $\mathbf{L}_p$ is defined as (p means power network)

$$(\mathbf{L}_p)_{ij} = \begin{cases} -c_{ij}, & i \neq j \\ \sum_{k\sim i} c_{ik}, & i = j \end{cases}, \tag{2.13}$$

and

$$\mathbf{B} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{L}_c \end{pmatrix}, \tag{2.14}$$

where $\mathbf{L}_c$ is defined as (p means communication network)

$$(\mathbf{L}_c)_{ij} = \begin{cases} -g_{ij}, & i \neq j \\ \sum_{k\sim i} g_{ik}, & i = j \end{cases}. \tag{2.15}$$

It is easy to verify that the matrices $\mathbf{L}_p$ and $\mathbf{L}_c$ are actually the Laplacian matrices of the power network and communications with weights $\{c_{ik}\}$ and $\{g_{ik}\}$ Chuang (1997). Laplacian matrix is ver y well studied in the graph theory field. It is a matrix representation of a graph and consists of a degree matrix and adjacency matrix. The elements of a Laplacian matrix are given as: 1) at the diagonal position, the elements value are the connected degree to all neighbors; 2) at the indices where the node has connected edge, the value is -1; 3) all the other positions have 0 value. Equation 2.16 below is an example of the Laplacian matrix of the graph in Figure 2.2:

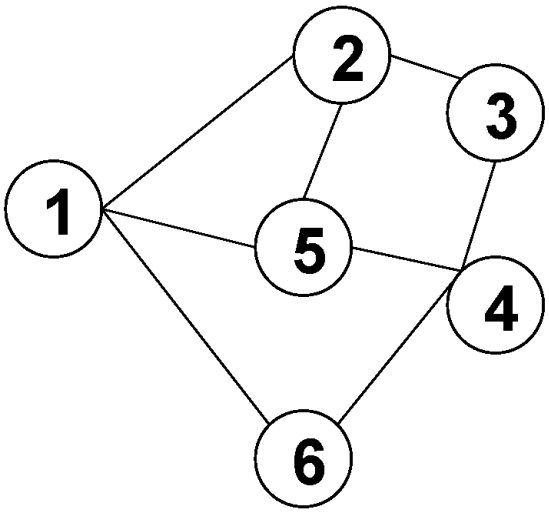

Figure 2.2: Example graph of the Laplacian matrix.

$$\mathbf{L} = \begin{pmatrix} 3 & -1 & 0 & 0 & -1 & -1 \\ 3 & -1 & 0 & 0 & -1 & -1 \\ 3 & -1 & 0 & 0 & -1 & -1 \\ 3 & -1 & 0 & 0 & -1 & -1 \\ 3 & -1 & 0 & 0 & -1 & -1 \\ 3 & -1 & 0 & 0 & -1 & -1 \end{pmatrix}, \tag{2.16}$$

We see that our $\mathbf{L}_p$ and $\mathbf{L}_c$ has the same structure as the example Laplacian matrix above. The eigenvalues of $\mathbf{L}_p$ and $\mathbf{L}_c$ have the following characteristics:

- All eigenvalues of $\mathbf{L}_c$ are non-negative since all the weights are positive. The smallest eigenvalue is 0.
- There is one zero eigenvalue for $\mathbf{L}_p$. The other eigenvalues may be positive or negative since the weights could be positive or negative.

2.1.5 Similarity to Flocking System in Control Field

A lot of studies in the control search field has studied the flocking system. Ferrari-Trecate et al. (2014) summarizes the major studies in the this filed. Based on Ferrari-Trecate et al. (2014), the interest here is in the coordination phenomena in applications such as controlling groups of unmanned autonomous vehicle. In such an application, each agent is usually described by a dynamical system characterizing the evolution of its position and velocity. Between different agents, information are shared through a communication network. Agent connected by communication link are considered as neighbors and the information from connected neighbors are immediately available. The analogy here of this control system to flocking system is each agent need to make moving decision, based on the input information from the communication network from connected agents in a decentralized way, just like each bird in flocking system need to make the moving decision based on the observation on the moving from neighboring birds. We can see the similarity here between our power grid system and this control system. Differently, in our power grid system, each power station is deciding on the frequency and phase, instead of the moving information based on the input from neighboring power station.

2.2 PdE Framework

I n this section, we fit the generator dynamics into the framework of PdE. We first assume that there is no communication delay and then extend to the case with non-negligible delay. Although the no-delay case is a special case of the latter, we can use it to better explain the framework of PdE. Then, we use arguments in the theory of flocking to analyze the system stability.

2.2.1 Field Operators in Networks

We first define various operators for the further analysis. These operators are the counterparts of operators in traditional field theory, in the context of discrete networks.

Gradient

First we define the gradient, which is of key importance in traditional field theory and PDEs. In the calculus over continuous space $\mathbb{R}^n$, the gradient of differentiable function $f(x_1, ..., x_n)$ at point $\mathbf{x} = (x_1, ..., x_n)$ is defined as

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}, ..., \frac{\partial f}{\partial x_n} \right), \tag{2.17}$$

where the partial derivative is defined as

$$\frac{\partial f}{\partial x_i} = \lim_{\delta \to 0} \frac{f(x_1, ..., x_i + \delta, ..., x_n)}{\delta}. \tag{2.18}$$

However, in the context networks, we cannot use the same definition of partial derivative since $\delta \to 0$ is not well defined in the discrete space of networks. However, we can capture the essential meaning of partial derivative, which means the changing rate between two 'neighboring' points with very small distance. Then, in the context of discrete networks, we can replace the traditional meaning of partial derivative with the change between to neighboring nodes. Motivated by this analogy, according to the framework of PdE in Ferrari-Trecate et al. (2014), we define the partial derivative of a vector function $\mathbf{r}$ mapping from a node a in a graph to $\mathcal{R}^2$ as

$$\partial_b \mathbf{r}(a) = \mathbf{r}(b) - \mathbf{r}(a), \tag{2.19}$$

where b is an adjacent node of a. Intuitively, we can consider node a as the 'anchor' point, and the edge between b and a as the direction or coordinate in the traditional calculus.

We can obtain the second order partial derivative as follows:

$$\begin{aligned} \partial_b^2 \mathbf{r}(a) &= \mathbf{r}(b) - \mathbf{r}(b) - (\mathbf{r}(b) - \mathbf{r}(a)) \\ &= -(\mathbf{r}(b) - \mathbf{r}(a)) \\ &= -\partial_b \mathbf{r}(a). \end{aligned} \tag{2.20}$$

We define the gradient of $\mathbf{r}$ as

$$\nabla \mathbf{r}(a) = \left(\partial_{b_1}^T \mathbf{r}(a), ..., \partial_{b_n}^T \mathbf{r}(a)\right)^T, \tag{2.21}$$

where b_1, ..., b_n are the neighbors of node a. Obviously, $\nabla \mathbf{r}(a)$ is the vector consisting of the partial derivatives of node a corresponding to its neighbors.

Divergence

For a $2n$-dimensional vector at node a, $\mathbf{s}(a) = \left(\mathbf{s}_{b_1}^T(a), ..., \mathbf{s}_{b_n}^T(a)\right)^T$, where each $\mathbf{s}_m$ is a 2-dimensional vector, we define its divergence as

$$\nabla \cdot \mathbf{s} = \sum_{j=1}^{n} \partial_{b_j} \mathbf{s}_{b_j}(i). \tag{2.22}$$

Difference and Similarity

We notice the following differences of the definitions of gradient and divergence in traditional continuous field and the discrete network under study:

- In traditional definition of gradient, the function is scalar, while we can define the gradient for vector functions in the context of discrete network.

- In traditional field theory, the divergence of a vector function is scalar, while the divergence defined here is 2-dimensional.

Despite the difference, both definitions in the context of discrete network are intuitively similar to those in traditional field theory, if we equalize the difference in network to the differential in continuous space.

2.2.2 PdE without Communication Delay

First, we assume $d = 0$, i.e., the communication delay is zero.

Diffusion Equation

Based on the definition of Laplacian in (2.21), we can consider the system state $\mathbf{x}$ as a set of 2-vectors over the set of nodes in a graph. Each node in the graph corresponds to a generator and the corresponding vector (say, for generator i) is $\mathbf{r}_i(t) = (\Delta\delta_i(t), \Delta f_i(t))^T$, namely the deviations of phase and frequency. To make the expression more compact, we add a virtual node, denoted by ϕ, with

$$\mathbf{r}_\phi(t) = \left(-\sum_{n=1}^{N} \int_0^t \Delta f_i(s) ds, 0 \right), \tag{2.23}$$

which is connected to all generators. Then, the linear dynamics in the vector form can be written as the evolution of vectors over a graph, which is given by

$$\frac{\partial \mathbf{r}_i(t)}{\partial t} = \nabla \cdot [\mathbf{\Psi}(i) \nabla \mathbf{r}_i(t)], \tag{2.24}$$

where $\mathbf{\Psi}(i)$ is a diagonal matrix coined the diffusion coefficient. $\mathbf{\Psi}(i)$ can be written as $\text{diag}(\mathbf{\Psi}_1(i), ..., \mathbf{\Psi}_n(i))$, where n is the number of neighboring nodes and each $\mathbf{\Psi}_k(i)$ is a 2×2 diagonal matrix. When i is not the virtual node and the neighbors are b_1, ..., b_n, $\mathbf{\Psi}_k(i)$ is determined by the following rules:

- When b_k is the virtual node, $\mathbf{\Psi}_k(i) = \begin{pmatrix} 0, & 1 \\ 0, & -D \end{pmatrix}$;
- When $b_k \sim i$ but $b_k \leftrightarrow i$ is not true (i.e., generator k is adjacent to i while there is no communication link between them), we have $\mathbf{\Psi}_k(i) = \begin{pmatrix} 0, & 0 \\ -c_{ik} & 0 \end{pmatrix}$.
- When $b_k \sim i$ and $b_k \leftrightarrow i$ (i.e., generator k is adjacent to i and there is a communication link between them), we have $\mathbf{\Psi}_k(i) = \begin{pmatrix} 0, & 0 \\ -c_{ik}, & -g_{ik} \end{pmatrix}$.

When i is the virtual node, we have $\mathbf{\Psi}_k(i) = \begin{pmatrix} 0, & -1 \\ 0, & 0 \end{pmatrix}$.

Note that (2.24) has exactly the same form as that of diffusion equation with homogeneous diffusion coefficients Crank (1975). The only difference is that the gradient and the divergence have new definitions in the context of dynamics over graph.

Property of Diffusion Operator

The dynamics of the power system in the above model are determined by the operator $\nabla \cdot [\mathbf{\Psi} \nabla]$, which is coined the diffusion operator. Similarly to Ferrari-Trecate et al. (2014), we define the following 'Sobolev' space:

$$H^1 = \left\{ f \middle| \sum_{i=1}^{N} f(i) = 0 \right\}. \tag{2.25}$$

Then, we have the following property of the operator $\nabla \cdot [\mathbf{\Psi} \nabla]$:

Lemma 2.0.1. *The operator $\nabla \cdot [\mathbf{\Psi} \nabla]$ maps from H^1 to H^1. Its eigenvalues are all negative if and only if the eigenvalues of the following matrix are negative:*

$$\mathbf{D} = \begin{pmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{L}_p & -\mathbf{L}_c \end{pmatrix}. \tag{2.26}$$

System Stability

Obviously, the origin point $\Delta\delta_i = 0$ and $\Delta f_i = 0$ (i.e., the frequencies equal the standard one) is a stationary point; however, its stability is unknown and is of key importance to the power grid. Following the definition of stability in Ferrari-Trecate et al. (2014), we say that the origin point $\Delta\delta_i = 0$ and $\Delta f_i = 0$ is stable if for all $t \geq 0$, for any $\epsilon > 0$, there exists a $\theta > 0$ such that

$$\|\mathbf{x}(0)\| \leq \theta \Rightarrow \|\mathbf{x}(t)\| \leq \epsilon. \tag{2.27}$$

Then, the following proposition discloses the sufficient and necessary condition of the stability of the equilibrium point of the power grid.

Proposition 1. *The origin point of the diffusion process (2.24) is stable if and only if the eigenvalues of the matrix in (2.26) have negative real parts.*

The proof is very similar to that of Theorem 4 in Ferrari-Trecate et al. (2014), which discusses the bird flocking. We can define a Lyapunov function and then apply Lemma 2.0.1 and Theorem 2 in Ferrari-Trecate et al. (2014) to prove that the Lyapunov function decreases with time. The details of the proof are omitted.

The following corollary provides a condition for the stability of the origin point in a special case.

Corollary 2.0.1. *Suppose that the Laplacian matrices $\mathbf{L}_c$ and $\mathbf{L}_p$ have the same eigenstructure and real eigenvalues. The origin point is stable if and only if the following inequality holds:*

$$\max_{0\leq\rho\leq 1} \max_{\|\mathbf{t}\|=1} -\rho \sum_{i=1}^{N} \lambda_i^c t_i^2 - \sqrt{\rho(1-\rho)} \min_i t_i(\lambda_i^p + 1) < 0. \tag{2.28}$$

where λ_i^c and λ_i^p are the eigenvalues of $\mathbf{L}_c$ and $\mathbf{L}_p$, respectively.

Proof. Using the variational expression of eigenvalues, we have

$$\lambda_{\max}^{D} = \max_{\|\mathbf{x}\|=1} \mathbf{x}^T \mathbf{D} \mathbf{x}, \tag{2.29}$$

where $\mathbf{D}$ is the matrix in (2.26). We decompose $\mathbf{x}$ as $(\mathbf{x}_1, \mathbf{x}_2)$. Then, we have

$$\begin{aligned} & \mathbf{x}^T \mathbf{D} \mathbf{x} \\ = \; & -\mathbf{x}_2^T \mathbf{L}_c \mathbf{x}_2 - \mathbf{x}_2^T \left(\mathbf{L}_p + \mathbf{I} \right) \mathbf{x}_1. \end{aligned} \tag{2.30}$$

Let $\|\mathbf{x}_1\|^2 = \rho_1$ and $\|\mathbf{x}_2\|^2 = \rho_2$. Obviously, we have $\rho_1 + \rho_2 = 1$. Transforming $\mathbf{x}_1$ and $\mathbf{x}_2$ such that $\mathbf{L}_c$ and $\mathbf{L}_p$ become diagonal, we have

$$\begin{aligned} & \mathbf{x}^T \mathbf{D} \mathbf{x} \\ = \; & -\sum_{i=1}^{N} \lambda_i^c x_{2i}^2 - \sum_{i=1}^{N} (\lambda_i^p + 1) x_{1i} x_{2i}. \end{aligned} \tag{2.31}$$

Fixing $\mathbf{x}_2$, we have

$$-\sum_{i=1}^{N} (\lambda_i^p + 1) x_{1i} x_{2i} \leq -\sqrt{1-\rho_2} \min_i (\lambda_i^p + 1) x_{2i}, \tag{2.32}$$

where the equality is achieved when

$$\mathbf{x}_{1i} = \begin{cases} \sqrt{1-\rho_2} & , \quad i = i^* \\ 0 & , \quad \text{otherwise} \end{cases}, \tag{2.33}$$

where $i^* = \arg\min_i (\lambda_i^p + 1) x_{2i}$. Hence, we have

$$\begin{aligned} & \mathbf{x}^T \mathbf{D} \mathbf{x} \\ \leq \; & -\sum_{i=1}^{N} \lambda_i^c x_{2i}^2 - \sqrt{1-\rho_2} \max_i (\lambda_i^p + 1) x_{2i}. \end{aligned} \tag{2.34}$$

Then, the conclusion is obtained by letting $\mathbf{t} = \frac{1}{\sqrt{\rho_2}} \mathbf{x}_2$. □

2.2.3 PdE with Communication Delay

Now we consider the case in which d is nonzero, i.e., the communication delay is non-negligible. Using the same argument as the no delay case, the dynamics of the power grid can be written as

$$\frac{\partial r(i,t)}{\partial t} = \nabla \cdot [\mathbf{\Psi}_p(i) \nabla r(i,t)] + \nabla \cdot [\mathbf{\Psi}_c(i) \nabla r(i,t-d)], \tag{2.35}$$

where the matrices $\mathbf{\Psi}_p(i)$ and $\mathbf{\Psi}_c(i)$ can be obtained similarly to the negligible delay case.

To study the stability of the PdE with communication delay, we need the following lemma (Lyapunov-Krosovskii Stability Theorem Gu et al. (2003)):

Lemma 2.0.2. *Consider a retarded functional differential equation (RFDE):*

$$\dot{\mathbf{z}} = f(t, \mathbf{z}_t), \tag{2.36}$$

where $f(t,0) = 0$. Suppose that there exist continuous functions u, v and w mapping from $\mathcal{R}_+$ to $\mathcal{R}_+$, which are nondecreasing. Moreover, $u(s)$ and $v(s)$ are positive for $s > 0$, and $u(0) = v(0) = 0$. If there exists a continuous differentiable functional V such that

$$u(\|\mathbf{z}(0)\|_2) \leq V(t, \mathbf{z}) \leq v(\|\mathbf{z}\|_c), \tag{2.37}$$

where $\|\mathbf{z}\|_c = \max_t \|\mathbf{z}(t)\|_2$, and

$$\dot{V}(t, \mathbf{z}) \leq -w\left(\|\mathbf{z}(0)\|_2\right), \tag{2.38}$$

then the trivial solution to (2.36), i.e., $\mathbf{z} = 0$, is uniformly stable. If $w(s) > 0$ for $s > 0$, the solution is uniformly asymptotically stable[‡].

[‡]A solution to (2.36) is uniformly stable if, for any t_0 and $\epsilon > 0$, there exists a $\delta(\epsilon) > 0$ such that $\|\mathbf{z}_{t_0}\|_c \leq \delta$ implies $\|\mathbf{z}(t)\| \leq \epsilon$ for all $t > t_0$. The solution is said to be uniformly asymptotically

According to Ordinary Differential Equations (ODEs), Lyapunov functions are scalar functions that can be used to prove the stability of an equilibrium of an ODE. The Lyapunov functions is a universal method for the investigation of the stability of nonlinear dynamical systems of general configuration. Informally, a Lyapunov function is a function that takes positive values decreases (or is non-increasing) along every trajectory of the ODE. Lyapunov function-based stability analysis of ODEs has the merit that the actual solution, analytical or numerical, of the ODE is not quired. Lyapunov (1992)

Then, we can prove the following conclusion about the property of the operator $\nabla \cdot [\mathbf{\Psi}_p(i)\nabla r(i,t)] + \nabla \cdot [\mathbf{\Psi}_c(i)\nabla r(i,t-d)]$.

Proposition 2. *The solution of (2.35) is uniformly stable if the eigenvalues of the following matrix are all nonpositive:*

$$\mathbf{F} = \begin{pmatrix} \mathbf{I} & 2\mathbf{I} \\ -2\mathbf{L}_p & \mathbf{L_c}^T\mathbf{L_c} + \mathbf{I} \end{pmatrix}. \tag{2.39}$$

If all eigenvalues of the matrix $\mathbf{F}$ *are negative, the solution is uniformly asymptotically stable.*

Proof. We define the following Lyapunov-Krasovskii functional:

$$\begin{aligned} V(\mathbf{x}(t)) &= \|\mathbf{x}(0)\|_2^2 + \|\mathbf{x}(t)\|_2^2 \\ &+ \int_{-d}^{0} \mathbf{x}^T(t+s)\mathbf{B}^T\mathbf{B}\mathbf{x}(t+s)ds. \end{aligned} \tag{2.40}$$

Obviously, we have

$$V(\mathbf{x}(t)) \leq 2\|\mathbf{x}(t)\|_c^2 + d\lambda_{\max}^*\|\mathbf{x}(t)\|_c^2, \tag{2.41}$$

stable if it is uniformly stable and there exists a $\delta > 0$ such that, for any $\eta > 0$, there exists a $T = T(\delta, \eta)$ such that $\|\mathbf{z}_{t_0}\|_c \leq \delta$ implies $\|\mathbf{z}(t)\| \leq \eta$ for all $t > t_0 + T$ Gu et al. (2003).

where $\lambda^*_{\max}$ is the maximum eigenvalue of $\mathbf{B}^T\mathbf{B}$ which is positive. Hence, we can set function v as the right hand side of (2.41). We can also set u as $u(s) = s^2$ such that $u(\|\mathbf{x}(0)\|_2) \leq V(t, \mathbf{x})$.

Taking the derivative with respect to t, we have

$$\begin{aligned}
\dot{V}(\mathbf{x}) &= 2\dot{\mathbf{x}}^T\mathbf{x} + \mathbf{x}^T(t)\mathbf{B}^T\mathbf{B}\mathbf{x}(t) \\
&- \mathbf{x}^T(t-d)\mathbf{B}^T\mathbf{B}\mathbf{x}(t-d) \\
&= 2\mathbf{x}^T\mathbf{A}\mathbf{x} + 2\mathbf{x}^T(t-d)\mathbf{B}\mathbf{x} \\
&+ \mathbf{x}^T(t)\mathbf{B}^T\mathbf{B}\mathbf{x}(t) - \mathbf{x}^T(t-d)\mathbf{B}^T\mathbf{B}\mathbf{x}(t-d) \\
&= 2\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T(t)\mathbf{B}^T\mathbf{B}\mathbf{x}(t) + \mathbf{x}^T\mathbf{x} \\
&- \big(\mathbf{x}^T\mathbf{x} - 2\mathbf{x}^T(t-d)\mathbf{B}\mathbf{x} \\
&+ \mathbf{x}^T(t-d)\mathbf{B}^T\mathbf{B}\mathbf{x}(t-d)\big) \\
&= 2\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T(t)\mathbf{B}^T\mathbf{B}\mathbf{x}(t) + \mathbf{x}^T\mathbf{x} \\
&- \|\mathbf{x} - \mathbf{B}\mathbf{x}(t-d)\|^2 \\
&\leq 2\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T(t)\mathbf{B}^T\mathbf{B}\mathbf{x}(t) + \mathbf{x}^T\mathbf{x}. \qquad (2.42)
\end{aligned}$$

Hence, if all eigenvalues of $2\mathbf{A} + \mathbf{B}^T\mathbf{B} + \mathbf{I}$ are nonpositive (negative), $\dot{V}$ is also nonpositive (negative). We can set

$$w(\mathbf{x}) = \lambda^{**}_{\max}\|\mathbf{x}\|_2^2, \qquad (2.43)$$

where $\lambda^{**}_{\max}$ is the maximum eigenvalue of $2\mathbf{A} + \mathbf{B}^T\mathbf{B} + \mathbf{I}$. Then, the conclusion is obtained from the Lyapunov-Krosovskii Theorem. □

We observe that the conditions for the stabilities are independent of the delay d. Hence, the conclusions is a delay-independent one Gu et al. (2003). Delay-independent stability condition has been studied for quite a while in control study community, e.g., Aleksandrov et al. (2014), Devane and Lestas (2015) and Ferrari-Trecate et al. (2014). In the case when delay is infinite, we can view the

communication link is down and there is no communication between two sub-stations. Such a situation leads to an open-loop control system, which is still possible to be stable. However, if the driving force of the dynamic system is a periodic one, for example, a sinusoidal function with a period of T, but the delay of the feedback control is $T/2$, the negative feedback will become positive feedback, which is even worse than no feedback (i.e. infinite delay). It is much more complicated to obtain a delay-dependent one, which involves the theory of linear matrix inequalities. As explained above, the Lyapunov function-based stability does not require an actual solution to the problem, but provides a theoretical way of deducting the stability condition of the problem.

2.3 Communication Topology Design

In this section, we study how to design the topology of the communication network in smart grid. In the case with communication delays, we need to optimize both the topology and delay, which is a complicated process and beyond the scope of the paper. To simplify the communication topology design problem, we consider only the case of no communications delays. Moreover, we assume that the feedback gain g_{ik} is a constant equaling g_0. It is more interesting to study the joint design of communication network and control law, which involves mixed integer programming and will be our future work.

2.3.1 Problem Formulation and Greedy Algorithm

As we have found, the stability of power system when there is no communication delay is determined by the eigenvalues of matrix $\mathbf{D}$. Hence, the objective could be minimizing the maximum eigenvalue of $\mathbf{D}$ in order to make all eigenvalues negative. Meanwhile, the communication network is subject to costs since the

communication links are not free. We denote by a_{ik} the cost of building the bi-directional communication link built between node i and k and assume that the total budget of the cost is a_{tot}. Then, the design of communication topology can be formulated as the following optimization problem:

$$\begin{aligned} & \min_{\{z_{i,k}\}_{i\neq k}} \quad \lambda_{\max}^{\mathbf{D}} \\ s.t. \quad & \sum_{i\neq k} z_{i,k} a_{ik} \leq a_{tot} \\ & z_{i,k} \in \{0,1\}, \qquad \forall i \neq k, \end{aligned} \tag{2.44}$$

where $z_{i,k}$ is the variable and means building (not building) the link between generators i and k when $z_{i,k} = 1$ (0).

Since the variable of the above optimization problem only takes 0 and 1 as solution to the variable, which is the representation of whether a communication link exists between two different power stations, the optimization problem above is an integer programming problem. More specifically, this problem is a binary constraint programming problem where only the 0 and 1 can be solution. The type of constraint programing originates from AI research. For example, in a chess board game optimization problem, symbolic values, e.g. positions on a chessboard, can be the variable to solve with the constraint of certain chess piece can only be assigned to certain position. Whether the position can have a chess piece or not is a 0-1 binary constraint optimization problem. Since the CP problems are non-convex, the solutions to solve such a problem usually involves certain exhaustive search algorithm, such as the ones used in the solver listed in Currie and Wilson (2012). In our communication topology design problem, assuming the power system has N power stations and they are completely connected through power lines, then the total number of decision variables in the communication network is then $\frac{N^2-N}{2}$, which shows the size of the problem is at the order of $O(N^2)$. Since each power line is being determined if there is a communication link between the connected power station, the complexity of the

problem is $O(4^N)$, which is a large number to solve in a binary integer programming problem, if N is large.

To alleviate the high computational time involved when using the traditional solver, we design a greedy algorithm to minimize the cost function of the above optimization step by step. The procedure is given in Procedure 1. In the algorithm, we try to add one communication link to the communication network in one iteration. In each iteration, we compute the maximum eigenvalue of all the possible matrices of adding that additional communication line, subject to the constraint that the communication line can only be added when there is power line but no communication line between two power station and when the total cost constraint is not violated. We then compare all the maximum eigenvalue with different communication topology and choose the one that has the minimum max eigenvalue to be the new communication topology with the new communication line. The program terminates when no more communication link can be added.

Algorithm 1 Procedure of Communication Network Topology Design

```
Initialize the communication links as an empty set.
while The total cost is less than a_tot do
    for Each communication link between i and j with z_ij = 0 do
        Set z_ij = 1 temporarily if there is power line between i and j.
        Compute the maximum eigenvalue of D.
    end for
    Choose the link with the minimum cost function.
    Set the corresponding z_ij = 1.
end while
```

2.3.2 Problem Relaxation with Semidefinite Program (SDP) Solution

The previous proposed type of greedy algorithm often gives a quick solution to the problem, compared with a solver that solves the entire problem rigorously. However, the heuristic algorithm does not give any performance guarantee. In this subsection,

we relax the optimization problem in last subsection to a problem that can be can be solved with well established solvers, such as SDP solvers, such that the computational complexity can be reduced, compared with solving the original problem with optimization solver.

Suppose λ_0 is the eigenvalue of matrix $\mathbf{D}$, with the corresponding eigenvector $[\delta^{\mathbf{T}}, \mathbf{f}^{\mathbf{T}}]^T$, where δ is the phase and $\mathbf{f}$ is the frequency vectors, we then have

$$\begin{aligned}\begin{pmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{L}_p & -\mathbf{L}_c \end{pmatrix} \begin{pmatrix} \delta \\ \mathbf{f} \end{pmatrix} &= \begin{pmatrix} \mathbf{f} \\ -\mathbf{L}_p\delta - \mathbf{L}_c\mathbf{f} \end{pmatrix} \\ &= \lambda_0 \begin{pmatrix} \delta \\ \mathbf{f} \end{pmatrix}.\end{aligned} \tag{2.45}$$

We then derive the following equations:

$$\mathbf{f} = \lambda_0 \delta \tag{2.46}$$

$$-\mathbf{L}_p\delta - \mathbf{L}_c\mathbf{f} = \lambda_0 \mathbf{f} \tag{2.47}$$

Hence, plugging Equation (2.46) into Equation (2.47), we have

$$\lambda_0^2 \delta = (-\lambda_0 \mathbf{L}_c - \mathbf{L}_p)\, \delta. \tag{2.48}$$

From Equation (2.48), we know δ is an eigenvector of $-\lambda_0\mathbf{L}_c - \mathbf{L}_p$, where its corresponding eigenvalue is λ_0^2. Therefore, the eigenvalue of $\begin{pmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{L}_p & -\mathbf{L}_c \end{pmatrix}$ is determined by the following equation

$$\lambda_0^2 = \widetilde{\lambda}\,(-\lambda_0 \mathbf{L}_c - \mathbf{L}_p) \tag{2.49}$$

where $\widetilde{\lambda}(\mathbf{A})$ denotes the eigenvalue of matrix $\mathbf{A}$.

Solving the above equation directly is non-trivial work. To simplify the problem, we make the assumptions that $\widetilde{\lambda}(\mathbf{L}_c) \leq 0$ and $\widetilde{\lambda}(\mathbf{L}_p) \leq 0$. We then have

$$\widetilde{\lambda}_2\left(-\lambda_0 \mathbf{L}_c - \mathbf{L}_p\right) \geq \lambda_0 \widetilde{\lambda}_2\left(-\mathbf{L}_c\right) \tag{2.50}$$

where $\widetilde{\lambda}_2(\mathbf{A})$ denotes the second largest eigenvalue of the the corresponding matrix. From Equation (2.50) we know a larger $\widetilde{\lambda}_2\left(-\mathbf{L}_c\right)$ can improve the lower bound of $\widetilde{\lambda}_2\left(-\lambda_0 \mathbf{L}_c - \mathbf{L}_p\right)$ and thus improve λ_0. Thus, we relax the communication topology design problem in Equation (2.44) to

$$\begin{aligned} & \max_{\{z_{i,k}\}_{i \neq k}} \quad \widetilde{\lambda}_2\left(-\mathbf{L}_c\right) \\ s.t. \quad & \sum_{i \neq k} z_{i,k} a_{ik} \leq a_{tot} \\ & z_{i,k} \in \{0,1\}, \qquad \forall i \neq k, \end{aligned} \tag{2.51}$$

where the $z_{i,k}$ is also the variable and means building (not building) the link between generators i and k when $z_{i,k} = 10$.

The problem formulation in Equations (2.51) can be easily derived to a convex optimization problem by further relaxing the second constraint above to a linear constraint, $0 \leq z_{i,k} \leq 1$. The solution of the convex relaxation is a superset of the our original problem, as it uses a linear constraint that includes the original discrete constraint. We refer to the similar formulation in Ghosh and Boyd (2006) and formulate such a convex relaxation to the following semidefinite program (SDP) problem

$$\begin{aligned} & \max \quad s \\ s.t. \quad & s\left(\mathbf{I} - (\mathbf{1}\mathbf{1}^T/n)\right) \preceq -\mathbf{L}_c \\ & \sum_{i \neq k} z_{i,k} a_{ik} \leq a_{tot} \\ & 0 \leq z_{i,k} \leq 1, \qquad \forall i \neq k, \end{aligned} \tag{2.52}$$

The above SDP programing problem can be solved with standard SDP solver, which usually finds solutions with a reasonable computational complexity.

2.4 Numerical Results

In this section, we evaluate the performance of different algorithms for solving the optimization problem discussed in this paper. The power network topology is randomly generated. We adopt the parameters of the transmission lines from the IEEE New England 39-bus model. The 39-bus model is a widely adopted model for testing new framework. It is a abstraction of a greatly reduced model of the power system in New England. There have been numerous research studies that adopt this model for both static and dynamic system research. The 39-bus system has 10 generators, 36 transmission lines, 19 loads and 12 transformers. The model is an AC power flow model, where real and reactive power flows and power system nonlinearity are characterized. In the simulation, we use the system topology of the power networks of the 39-bus model as the base and design the communication network accordingly. The system model is shown in Figure 2.3.

2.4.1 Greedy Algorithm Performance

We first evaluate the performance of the greedy algorithm proposed in Procedure 1. The power network is composed of 9 power generators with a randomly generated power network topology. The algorithm is applied for several settings of g. Fig. 2.4 shows the change of the cost function, the maximum eigenvalue of the matrix D, as more communication links are added, when the greedy algorithm in Procedure 1 is applied. We observe that the real part of the maximum eigenvalue is still positive, although it is close to zero. Although Theorem 1 gives the sufficient and necessary condition for the power system to be stable, it is the condition at both diffusion origin, where both the frequency component Δf and the phase component $\Delta\delta$ is

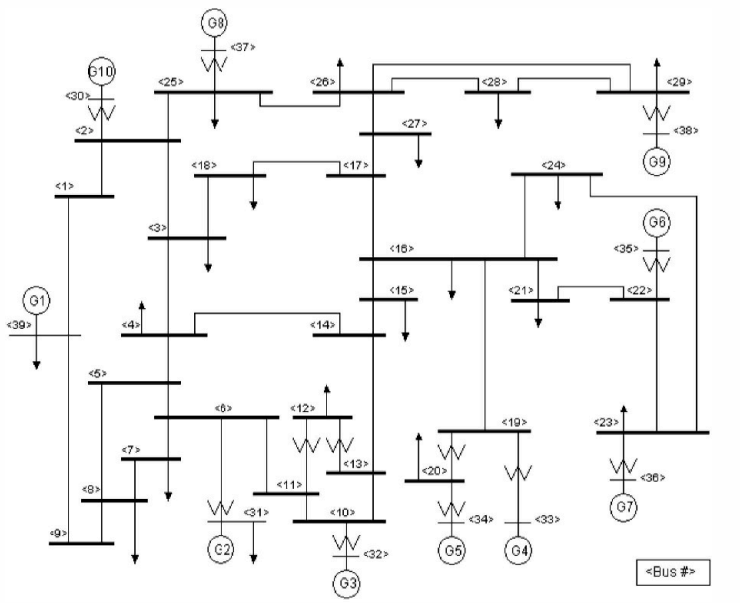

Figure 2.3: Illustration of the 39-bus model.

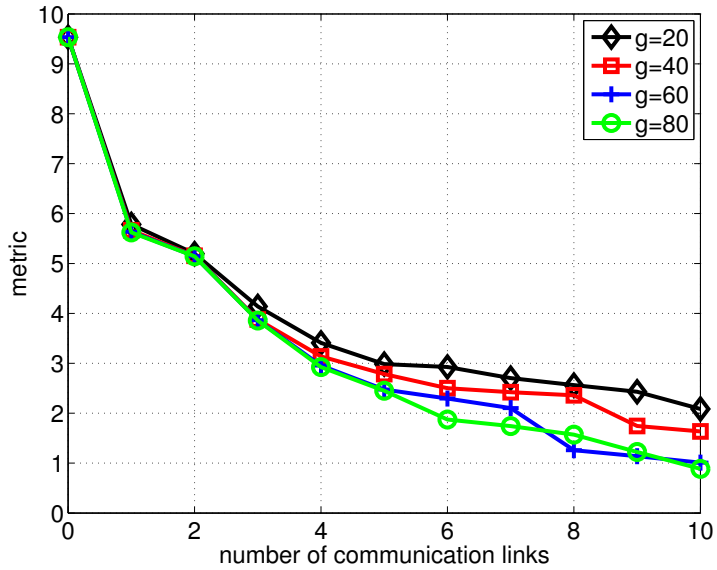

Figure 2.4: Evolution of the metric obtained in Procedure 1.

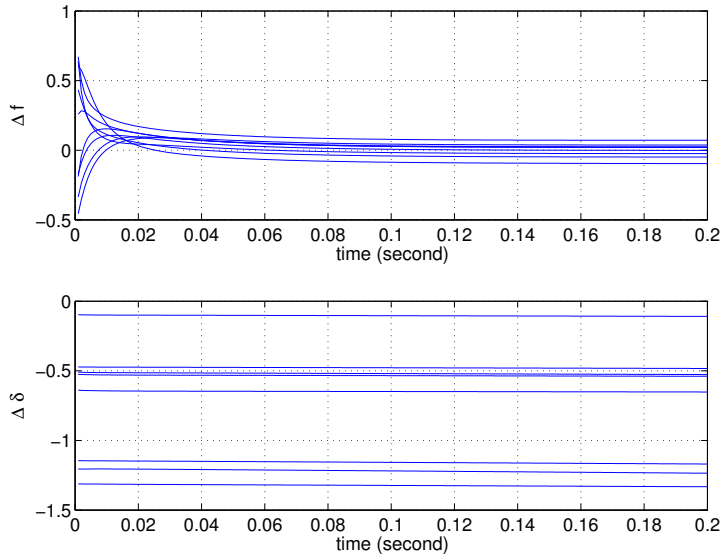

Figure 2.5: Evolution of Δf and $\Delta\delta$.

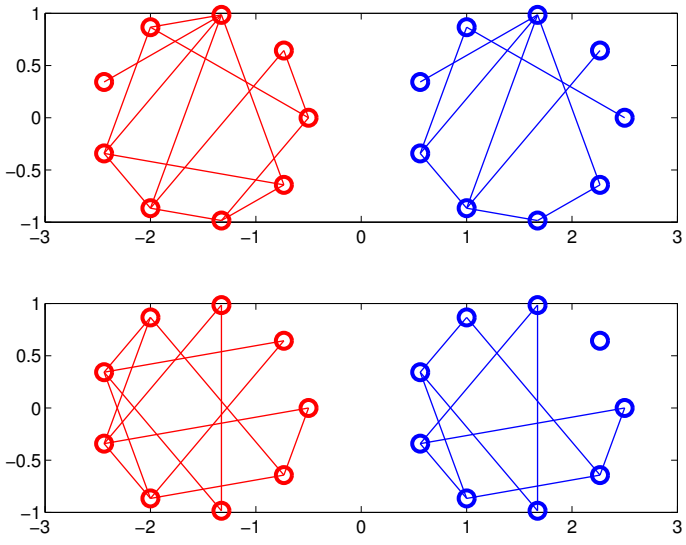

Figure 2.6: Illustration of the power and communication networks.

zero. However, due to the existence of phase, the indifference of phase change and that the phase component $\Delta\delta$ cannot be recovered to zero since it is accumulated over time, the origin of the phase and frequency actually cannot be stable. Hence we tried the conclusion in that theorem that is actually not precisely applicable to the numerical problem, which leads to the real part of the eigenvalue to be still positive. It is similar to the belief propagation in machine learning, which behaves well but lacks a complete theoretical explanation.

Nevertheless, we observe that the cost function is substantially decreased, which shows the trend that the system is converging to stable with more number of links added to the communication topology Fig. 2.5 shows that the frequency deviation converges to zero (the upper figure) while the phase deviations remain almost constant. This implies that a new criterion is needed to characterize only the convergence of Δf to 0 if it is not necessary to force the phase deviations back to zero. If the system requires to force the phase deviation back to zero as well, it is necessary to introduce the phase measurements into the control action, similarly to the control law in Li and Han (2011). The numerical results show that it works numerically although there is still no theoretical explanation for it. Two examples of the power network topology (left) and the optimal communication network topology (right) are shown in Fig. 2.6. We see that the algorithm can pick links from the power network and make the communication network.

2.4.2 Greedy Algorithm Vs. Optimization Solver

In this section, we compare the performance of the proposed greedy algorithm and the existing optimization tool package. From the previous analysis we know that the complexity of the problem is in the order of $O(4^N)$, where N is the number of power stations. It is therefore an exponential order mixed integer programming problem. To solve the problem, we use existing optimization solver in this evaluation section. The optimization package we use to solve this problem is the OPTimization

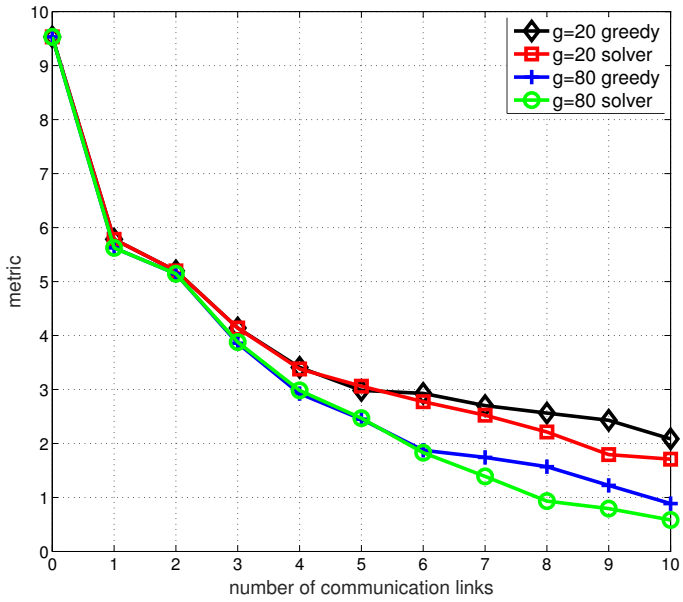

Figure 2.7: Evolution of the optimized cost using greedy algorithm and OPTI optimization sober.

Interface (OPTI) Toolbox, designed for constructing and solving linear and nonlinear continuous and discrete optimization problems Currie and Wilson (2012). It utilizes open source solvers to solvers such as IPOPT, SCIP, etc., to solve the problem. Our problem is a mixed integer nonlinear programing problem, which can be solved using the OPTI toolbox.

The power topology used in the evaluation is the same nine generators system with randomly generated power link topology. Figure 2.7 shows the change of the cost function when the number of the communication links increases with different algorithms. We also evaluate the performance using different g values. From the results we see that the optimization performance of the OPTI solver outperforms the greedy algorithm only when the number of communication links is high. And the performance gain is not significant. However, in our evaluation, the OPTI solver takes significantly more time to solve the optimization problem as it searches through the entire possible solution space to find an optimization solution. When

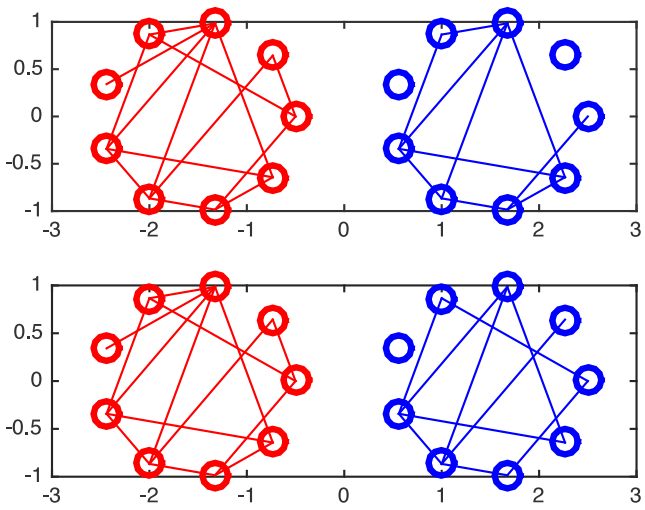

Figure 2.8: Communication graph (blue) design when $g = 20$. The top two is the solution from the greedy algorithm. The bottom row is the solution from the OPTI package.

the number of communication links is high, the possible solution space is also large, which makes the OPTI solver slow when solving the problem. Figure 2.8 and 2.9 show the communication graph (in blue) of the different algorithms when $g = 20$ and $g = 80$, respectively. In each graph, the top row is the solution from the greedy algorithm, while the bottom row is the solution from the OPTI solver. We see that the communication links are more evenly distributed among different stations in the solution from the OPTI solver. Moreover, we see that the greedy algorithm gives a solution that leaves two power station disconnected in the communication network, while the OPTI solver solution has a better connection in communication network among the power stations.

2.4.3 Performance of Greedy Algorithm vs. SDP optimization

In section 2.3.2, we relaxed the optimization problem of solving the communication topology to an SDP optimization problem. Here, we evaluate the optimization results of the original problem solved with the greedy algorithm and the relaxed

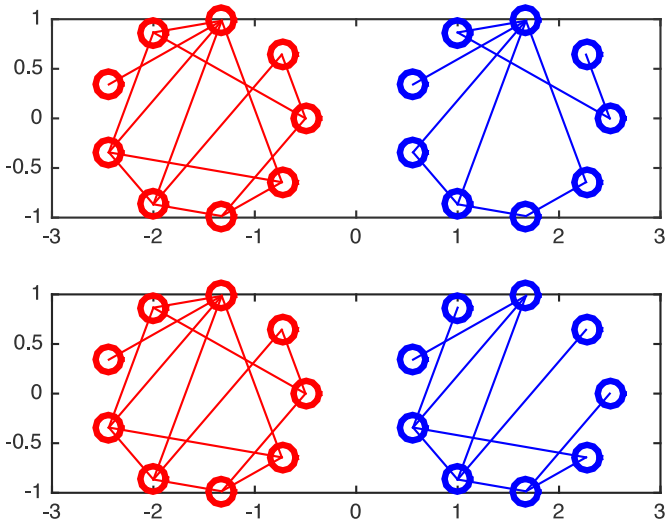

Figure 2.9: Communication graph (blue) design when $g = 80$. The top two is the solution from the greedy algorithm. The bottom row is the solution from the OPTI package.

SDP optimization problem solved by existing SDP optimization package. The SDP optimization package we use is CVX, a Matlab based modeling system for convex optimization Grant and Boyd (2014)Kundur (2008).

We also change the system topology in this evaluation experiment. Different from the previous 9 power station system, we use a 15 power station system derived from the IEEE New England 39-bus model. In the evaluation, the maximum total number of communication links is increased to 20. The power link topology is randomly assigned. Figure 2.10 shows the change of the cost function when increasing the number of communication links. We see that compared with solution to the original optimization problem solved with the greedy algorithm, the SDP variant of the original optimization problem shows a higher cost function optimization result. The reason is that the SDP relaxation aims to optimize $\widetilde{\lambda}_2(-\mathbf{L}_c)$, which only considers the communication matrix. Without the information from the power matrix, the optimization results can not reach a lower cost result. However, an SDP problem is often easy to solve and provides a theoretical boundary of the optimization results. The complexity of the solving the SDP problem is much lower than the previous

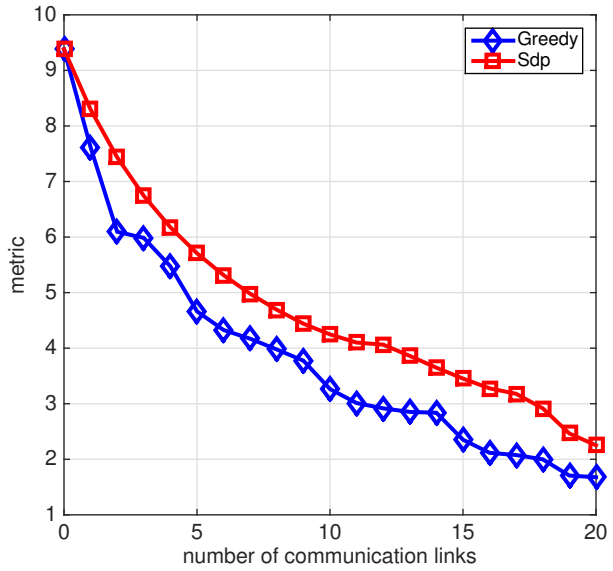

Figure 2.10: Evolution of the optimized cost using greedy algorithm and the cost optimization for the SDP relaxation problem.

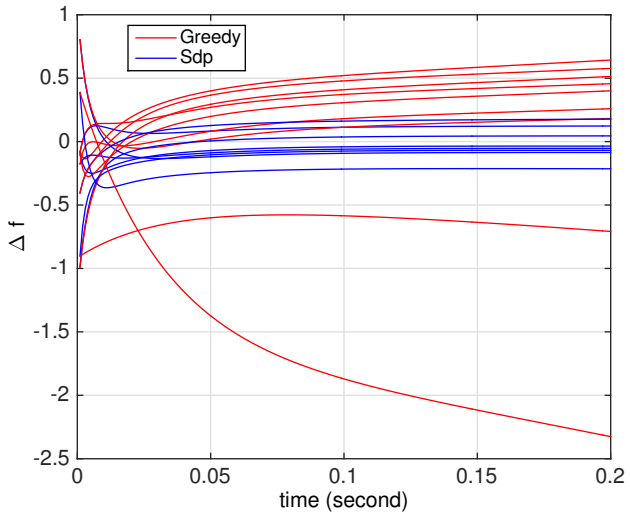

Figure 2.11: The frequency deviation convergence.

solution with optimization solver. Therefore, the SDP variant provides a fast solution to get a boundary solution of the optimization problem. Figure 2.11 shows the frequency deviation convergence, where the SDP problem has a faster convergence speed.

Chapter 3

End-to-end Delay Control with Power Efficient Wireless Sensor Networks

Power systems has three major sub-subsystems, which are power generation, power delivery and power utilization. Because of the low-cost and non pervasive nature of wireless sensor networks, recently WSNs have been identified as a promising technology to be utilized in all of the three subsystems. WSN is considered as a key enabler to the future smart grid system. WSN, if utilized effectively, can provide efficient monitoring the critical smart grid equipment, as well as monitoring and responding to the change conditions in the smart grid with a proactive manor Tuna et al. (2013).

Here, we adopt wireless sensor network (WSN) as the infrastructure of the communication network. To effectively use the WSN technology for power grid monitoring, we need to provide certain the Quality of Service in the WSN infrastructure. Since the wireless sensor networks has limited power, the transmission range of a wireless sensor node is usually limited. To apply WSN in an application such as the power grid, information transmission between two end nodes are usually relayed by multiple

different nodes in between. Therefore, in this chapter, we focusing on solving the problem of end-to-end delay control of the WSN communication network, such that it can be better utilized for power grid application. In addition to the end-to-end delay control problem, we also seeks to prolong the life-time problem of WSN, as the lifetime of the WSN infrastructure is a major concern when deploying it in real-world applications.

3.1 Literature Study

Several protocols have been proposed to provide delay guarantees for wireless sensor and ad hoc networks. Implicit EDF Caccamo et al. (2002) is a collision-free scheduling scheme which provides delay guarantees by exploiting the periodicity of WSN traffic. RAP Lu et al. (2002) uses a velocity monotonic scheduling scheme to prioritize real-time traffic based on a packet's deadline and its distance to the destination. SPEED He et al. (2003) achieves end-to-end communication delay guarantees by enforcing a uniform communication speed throughout the network. Karenos et al. Karenos and Kalogeraki (2006) have also presented a flow-based traffic management mechanism to provide delay guarantees. Our work is different from the aforementioned research. By dynamically manipulating the sleep interval, we provide delay guarantees for end-to-end communications, while the delay incurred by sleeping nodes is not considered in the aforementioned protocols.

Periodic sleeping is a widely adopted approach to saving energy for WSNs. The existing periodic sleeping approaches can be categorized into two classes: *static sleep scheduling* and *dynamic sleep scheduling*. In the static sleeping approach category, S-MAC Ye et al. (2002) proposes a synchronous periodic sleeping MAC with fixed duty cycles for energy savings. D-MAC Lu et al. (2004) is developed especially for a tree topology network. It aims to reduce sleep latency while decreasing energy consumption. Several other static sleep scheduling protocols (e.g., van Dam and Langendoen (2003)Ha et al. (2006)) are also proposed. However, none of the above

studies provides delay guarantees when utilizing periodic sleeping to save energy. A static sleep scheduling approach with delay guarantee has been recently proposed Gu et al. (2009). However, it cannot adapt to network condition changes such as interference incurred by additional workload at runtime. The second class of periodic sleeping schemes is dynamic sleeping scheduling. In those approaches, nodes are allowed to have different sleep schedules and change their schedules dynamically at runtime. Min et al. Min et al. (2008) propose to choose different nodes to go to sleep at different time based on the Analytic Hierarchy Process (AHP). Ning et al. Ning and Cassandras (2008) propose to use the dynamic programing approach to control sleep time of nodes for energy minimization. Different from the existing dynamic sleep scheduling approaches, our work dynamically adjusts the sleep interval of each node based on the delay constraint and the network condition changes.

The control-theoretic approach has been applied to various computing and networking systems. A survey of feedback performance control for software services is presented in Abdelzaher et al. (2003). However, only a few recent studies in wireless sensor networks start to utilize feedback control theory to provide performance guarantees. ATPC Lin et al. (2006) employs a feedback-based transmission power control algorithm to dynamically maintain individual link quality over time in WSNs. Merlin et al. Merlin and Heinzelman (2008) propose to control the duty cycle of each node for the desired throughput in a WSN. A control-theoretical approach is also designed in Le et al. (2007) to achieve the maximum network throughput in multi-channel WSN. To our best knowledge, this work is the *first* one that takes advantage of feedback control theory to dynamically control the sleep interval of every individual node for end-to-end delay guarantees in WSNs.

3.2 Problem Formulation

In this section, we introduce the formulation of our problem. We assume that the network is composed of m sources, some relay nodes and multiple sinks. A data

flow in the network includes a source to generate data packets based on a certain distribution, multiple nodes to relay the packets hop by hop, and a corresponding sink node. There are m flows in the network, each of which is generated by one of the m sources. All flows are assumed to be disjoint with each other because disjoint flows are widely used in multi-path routing to enhance the system's fault-tolerance Wang et al. (2009b)Maimour (2008). However, our approach can be easily extended to the case where multiple flows share the same nodes by allowing the shared nodes to wake up at all the wake-up time instants decided by different flows.

We assume that nodes operate in a periodic sleep schedule where each sleep period consists of a *sleep interval* and a *wake-up interval*. The *sleep interval* is the time duration when the node's radio is off in each sleep period. The *wake-up interval* is the time duration that a node has its radio on to transmit packet. Our primary goal is to design a dynamic duty cycle control policy to dynamically tune the sleep interval of each node so that the communication on each data flow can achieve an end-to-end delay guarantee while taking advantage of periodic sleeping to save energy. We first introduce the following notation:

- $G = (E, V)$, a WSN where V is the node set and E is the communication link set of the network.
- $f(u_0, u_n)$, a node set which forms a single data flow with source u_0 and destination u_n. The node index in a flow is enumerated from 0 to n in the hop sequence. Specifically, $f(u_0, u_n)$=$\{u_i|(u_{i-1}, u_i) \in E, 1 \leq i \leq n\}$.
- F, the set of all m flows. Specifically, $F = \{f_j|\forall j : 1 \leq j \leq m\}$
- d_i, the single-hop communication delay on link (u_{i-1}, u_i), which is formally defined in Section 3.3.
- $D_{ref}^{f_j}$, the end-to-end delay requirement of flow f_j.
- c_i, the time length of the sleep interval in one sleep period of node u_i.

Our goal is to control the end-to-end communication delay according to a given end-to-end delay requirement for each data flow. We formulate this problem as follows:

$$\min_{\forall f_j \in F} \left| \sum_{u_i \in f_j} d_i - D_{ref}^{f_j} \right| \tag{3.1}$$

However, to solve the problem defined above, one must know the global information of a flow in the WSN, such as the communication delay of each hop. This is inefficient, especially when flows have high hop counts. As our goal is to achieve the end-to-end delay control, we decompose our end-to-end delay control problem into a set of single-hop delay control subproblems. By achieving the single-hop delay control goal, we can control the multi-hop end-to-end delay. Specifically, for each flow f_j, our objective is:

$$\min_{\forall u_i \in f_j} \left| d_i - D_{ref}^{i} \right| \tag{3.2}$$

subject to the constraints

$$\sum_{u_i \in f_j} D_{ref}^{i} = D_{ref}^{f_j} \tag{3.3}$$

$$d_i \geq D_{min} \tag{3.4}$$

$$c_i \geq 0 \tag{3.5}$$

where D_{ref}^{i} is the single-hop delay requirement for link (u_{i-1}, u_i) and D_{min} is the minimum transmission time for a packet to be successfully received by the receiver. Constraint (3.3) enforces the single-hop delay requirement based on the end-to-end delay requirement. Constraint (3.4) means that the single-hop transmission delay has a lower bound, which is decided by the transmission rate of the radio and the packet size. Constraint (3.5) enforces that sleep interval of any node must be non-negative.

3.3 Single-hop Delay Control

In this section, we first present a single-hop delay model. Our model characterizes the expected one-hop communication delay by taking into account several realistic factors, such as network conditions and retransmission delays due to lossy links. Based on the model, we introduce the design of the single-hop delay controller.

3.3.1 Single-hop Delay Model

We assume that after packets are received by a node, they are immediately ready for being transmitted to the next hop without a queuing delay. This assumption is relaxed in Section 3.4. The sender is assumed to know the sleep schedule of receiver. We will discuss in Section 3.3.4 how this can be achieved. We also assume that the sender will try to send the packet only once every time the receiver wakes up. If the packet is not successfully received by the receiver during the wake-up time of the current sleep period, the sender will go to sleep and try to send the packet at the receiver's wake-up time in the next period. This prevents a sender from keeping using the channel for a long time, so that other nodes cannot get the channel during their wake-up times, especially when the link quality is low. Thus, the time delay, $d(k)$, for the kth packet to transmit from the sender to the receiver can be modeled as:

$$d(k) = \frac{c(k-1) + t_{data}}{PRR(k)} \tag{3.6}$$

where $c(k-1)$ is the sleep interval to be used at the receiver after the $(k-1)$th packet is received. $PRR(k)$ is the average packet reception ratio estimated when the kth packet is ready for transmission. It is a metric widely used to quantify the quality of links Woo et al. (2003). t_{data} is the time needed to transmit one packet after getting the channel, which includes processing time and the time to transmit the packet on radio. It can be approximated as a constant because the packet size and transmission rate usually do not change and it is significantly smaller than the sleep period. Since

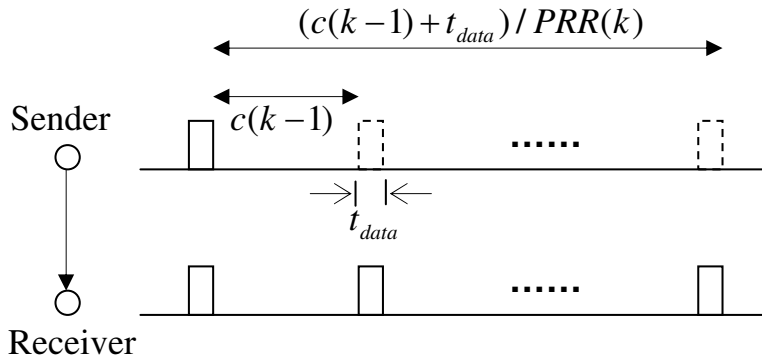

Figure 3.1: Single-hop delay is the time length of total sleep periods used to successfully transmit a packet.

we do not use CSMA, t_{data} does not include a backoff time. Contention interference is captured in $PRR(k)$.

Figure 3.1 illustrates the single-hop delay model in Equation (3.6). The total single-hop delay is the number of transmissions it takes to successfully transmit the packet multiplied by the time needed for each sleep period, which is the summation of the sleep interval and the wake-up time to transmit a packet. In the case that the packets arrive aperiodically, we can simply add a time difference quantity to Equation (3.6), which is defined as the time difference between the packet arrival time and the closest wake up time of the sender.

We assume that the average network condition of a link does not change frequently, compared with the wake-up frequency of the node on that link. In this case, PRR remains constant between the arrivals of two back-to-back packets. We assume that the PRR values of the links in our network are higher than a threshold in order to reach the communication link requirement Woo et al. (2003). Zhao et al. Zhao and Govindan (2003) show that the PRR value is temporally stable when it is relatively high. Therefore, we simplify the PRR value to be a constant during the transmission of two consecutive packets. Using Equation (3.6), we can derive the dynamic model of our single-hop communication delay as a difference equation in Equation (3.7), where $\Delta c(k) = c(k+1) - c(k)$.

$$d(k+1) = d(k) + \frac{\Delta c(k)}{PRR} \tag{3.7}$$

In the above model, PRR is the estimated average success rate of the packet transmission on a single link. In different areas of the network, we may have different values of PRR due to network condition variations. In order to capture the uncertainty of the network condition, we add an uncertainty ratio to our model as:

$$d(k+1) = d(k) + g\frac{\Delta c(k)}{PRR} \tag{3.8}$$

where g represents the ratio between the estimation of PRR and the actual PRR under the current network condition due to the uncertainty of the network environment. For example, $g = 2$ means that the actual packet reception ratio of the current network is half of the estimated packet reception ratio. Note that the exact value of g is unknown at design time due to the unpredictable network condition. We explain how we handle this uncertainty in the next subsections.

3.3.2 Feedback Controller Design

The core of any feedback control loop is the controller. In each control period, the controller monitors and controls a *controlled variable* by adjusting a system parameter, called *manipulated variable*, in order to meet a system requirement, usually called *performance reference*. In our problem, we try to control the single-hop delay of each packet to meet the delay requirement by dynamically adjusting the receiver's sleep interval. Therefore, the *controlled variable* in our problem is the single-hop communication delay of the next packet. The *manipulated variable* is the sleep interval time that the receiver sets for the next packet and the performance reference is the single-hop delay requirement, denoted as D_{ref}.

Note that the model in Equation (3.8) cannot be directly used to design the controller because the g is used to model the uncertainties in the network conditions and thus, is unknown at design time. Therefore, we design the controller based on

an approximate system model, which is model (3.8) with $g = 1$. In a real network where the packet reception ratio is different from the estimation, the actual value of g may be different than 1. As a result, the closed-loop system may behave differently. However, in next subsection, we show that a single-hop delay controlled by the controller designed with $g = 1$ can remain stable as long as the variation of g is within a certain range. This range is established using a stability analysis of the closed-loop system by considering model variations.

Following standard control theory Franklin et al. (1997), we design a Proportional (P) controller to achieve the desired control performance, such as stability. We can derive the receiver's desired sleep interval for the kth packet as shown in Equation 3.9.

$$c(k) = (D_{ref} - d(k)) \times PRR + c(k-1) \tag{3.9}$$

It is easy to prove that the controlled system is stable and has zero steady state errors when $g = 1$. The detailed proofs and design procedures can be found in a standard control textbook Franklin et al. (1997). As shown in Equation 3.9, the computational overhead of the P controller is just two additions/subtractions and one multiplication and is thus small enough to be implemented in a real sensor mote.

3.3.3 Stability Analysis for PRR Variation

In this section, we analyze the system stability when the designed P controller is used in an area where $g \neq 1$. A fundamental advantage of the control-theoretic approach is that it provides confidence for system stability, even when the packet reception ratio deviates from the estimation.

The closed-loop transfer function for the real-system is:

$$G(z) = \frac{g}{z - (1 - g)} \tag{3.10}$$

The closed-loop system pole in Equation (3.10) is $1 - g$. In order for the controller to be stable, the pole must be within the unit circle. Hence, the system will remain

stable as long as $0 < g < 2$. The result means that despite every link may have a different g, in order to achieve stability, the estimated PRR of a link should be less than twice its actual PRR. In WSN applications, we usually have a lower threshold of PRR, which is the lowest PRR that can provide an acceptable communication quality Woo et al. (2003). If the actual PRR of a link is less than the threshold, we simply cannot use this link for communication. As our goal is to dynamically control the sleep interval to achieve the delay guarantee, we assume that the links in the control problem are communication links. Thus, with a lower threshold of PRR, we can have the stability guarantee. For example, if the threshold of PRR is 0.5 which indicates that a packet is retransmitted twice on average before it is successfully received, we can use any estimated PRR value in the controller for stability, since the estimated PRR is always less than 1. Detailed empirical studies on link quality and PRR can be found in Woo et al. (2003).

3.3.4 Implementation

A periodic sleeping scheme requires the sender to be aware of the receiver's sleep schedule, so that the sender can also wake up to transmit packet when the receiver wakes up. This means that the information of the sleep interval change of the receiver needs to be shared by the sender. Therefore, we implement the controller on the receiver side and take advantage of the ACK packet to feed the updated sleep interval back to the sender side.

On the sender side, the sender first timestamps the packet when the packet is received (or generated if the sender is source). Then the sender will add the transmission times of the current packet in the packet header. This information is updated every time the packet is being transmitted (or retransmitted). Upon successfully receiving the packet, the receiver calculates the time delay based on the time stamp on the sender side and runs the controller to compute the new sleep interval for the receiver's sleep scheduling, starting from the next packet. The receiver then inserts the newly updated sleep interval value in the ACK packet and sends

the ACK back to the sender. The sender updates the receiver's sleep scheduling information on its own side for future transmissions. To handle the loss of ACK packets, we further adopt a localized synchronization scheme. Whenever the sleep schedule is successfully updated at both the sender and receiver through an ACK, we set up a timer with an interval several times longer than the duty cycle. If the sleep schedule is not updated during this interval, the sender and receiver wake up themselves and synchronize their schedules.

3.3.5 Integration of Multiple Single-hop Controllers in Tree-based Topology

The formulation in Section 3.2 is based on the topology that each end-to-end flow is disjoint with other flows in the network. In this type of network, each relay node only needs to serve a single sender. As we have introduced in Section 3.2 that tree-based topology is another widely adopted topology in WSN applications like data collection, etc. In tree-based topology, each relay node serves more than one senders such that flows generated from different sources can share a same relay node for packet relaying.

Since each flow may have different end-to-end delay requirements, to serve more than one sender, a relay node needs to schedule its own sleeping scheduling such that the single-hop delay requirement of all the incoming links can be guaranteed. One approach to design the feedback control sleep scheduling mechanism for this multiple incoming links situation is to design a single sophisticated multi-input single-output controller at the receiver, such that the delay of multiple incoming links can be considered together. However, a problem with this approach is when the receiver decide to use a sleep schedule, it cannot take advantage of the ACK packet to send the new schedule back. Instead, it needs to generate new packets and send them to all the senders with the new schedule information. This phase increases the transmission overhead of the network, which leads to undesired delay and energy consumption. In our approach, we simply allow each node to maintain several sleep schedules at the

same time, one for each incoming link. The shared node only goes to sleep when all the sleep schedules are in the sleep state. The shared node wakes up when any one of the sleep schedule needs to wake the node up. Therefore, although a relay node with multiple incoming links does not show periodic sleeping behavior overall, it actually follows periodic sleep scheduling in terms of each incoming link.

3.4 Queuing Delay Adaptation

In the last section, we assume that the incoming packet rate on the sender side is low, such that the packet is immediately available for transmission without a queuing delay. However, the packet receiving time (or generating time) at the sender is usually aperiodic and unpredictable, especially in event-driven WSN applications. For example, in surveillance applications for a battlefield, packets are generated upon the detection of intruders in the monitored area, which is usually unknown a priori. If the packet generation rate is high in such applications, the transmission of the packet may suffer additional time latency because of the queuing delay at the sender. With the queuing delay, the model of our transmission delay in Equation (3.6) is subject to changes. In this section, we consider the queuing delay caused by the unpredictable packet rate when adjusting the sleep interval for the next packet.

3.4.1 Impact of Queuing Delay

As introduced previously, when the incoming packet rate is high, the packet may suffer additional delay from queuing on the sender side because of the busy MAC layer. Therefore, we need to consider the queuing delay in our single-hop delay model. The model from Equation (3.6) is changed as follows

$$D(k) = t^q(k) + \frac{c(k-1) + t_{data}}{PRR(k)} \tag{3.11}$$

where $t^q(k)$ is the queuing delay of the kth packet at the sender and $D(k)$ is the total delay when the queuing delay is counted. Compared with the queuing delay, the delay incurred by the upper layer for computation purpose is small and negligible. Therefore, the queuing delay can be calculated as the time difference between the moment the packet is received and the moment the packet is ready to be sent in the MAC layer. In essence, the queuing delay of the jth packet in the queue is the time needed to transmit all the packets that are ahead of packet j in the queue. With the assumption we made in Section 3.3.1 that the network condition does not change frequently, the queuing delay for the jth packet in the queue can be calculated as:

$$t^q(j) = \sum_{n=1}^{j-1} d(n) = \sum_{n=1}^{j-1} \frac{c(n-1) + t_{data}}{PRR} \tag{3.12}$$

From Equation (3.12), we know that the queuing delay of the jth packet in the queue depends highly on the delay of all the packets ahead of it in the queue. The goal of our design is to choose a sleep interval at the receiver such that the single-hop delay for the next packet is controlled to the reference D_{ref}. Therefore, when calculating the sleep interval at the receiver for the next packet, we need to consider its impact on the queuing delays of all the packets in the queue on the sender side.

Suppose that the new sleep interval $c(k)$ will be used until the queue on the sender side empties. We can calculate the queuing delay of any packet $(k+n)$ (the nth packet currently in the queue) as:

$$t^q(k+n) = \frac{n(c(k) + t_{data})}{PRR} \quad \forall n : 1 \leq n \leq j \tag{3.13}$$

In order to achieve energy efficiency, we need to choose a maximum sleep interval $c(k)$ under the constraint that the new sleep interval will not cause any packet in the queue to violate the delay requirement. We denote the slack time left for the $(k+n)$th packet until it expires, based on the single-hop reference D_{ref}, as $T_{slack}(k+n)$. The

formulation of this sleep interval calculation is as follows:

$$\max \ c(k) \tag{3.14}$$

subject to the constraint:

$$\begin{aligned} T_{slack}(k+n) \geq & t^q(k+n) + d(k+n) \\ & \forall n : 1 \leq n \leq j \end{aligned} \tag{3.15}$$

where j is the total packet number in the queue after the kth packet.

3.4.2 Implementation and Coordination with Single-Hop Delay Controller

We now discuss the details of the implementation for the sleep interval calculation when we need to adapt to the queuing delay. Similar to the single-hop delay controller designed in Section 3.3, we implement the new sleep interval computation on the receiver side. The difference is that the sender will first compute a minimal *nominal sleep interval* only based on the slack time of each packet in the queue, without considering the network condition. The *nominal sleep interval* is added to the first packet in the queue, which is the next packet to be transmitted. Upon receiving the packet, the receiver will calculate the real desired sleep interval for the next packet, based on the current network conditions. After the calculation of the new sleep interval, the receiver uses the ACK packet to feed the sleep interval change back to the sender.

Since we have two approaches that both dynamically adjust the sleep interval at the receiver: feedback delay control and queuing delay adaptation, a coordination scheme must be designed in order to avoid any conflicts. In our design, we use queuing delay adaptation when the queue at the sender is not empty. The reason is that the delay controller controls the delay of every packet. If we use the delay controller

to perform feedback control for every packet when there are packets waiting in the queue, the packets in the queue after the current packet are likely to miss their delay requirements. This is because that the delay controller does not consider the queuing delay incurred by the unpredictable incoming packet rate. If there is no packet in the queue when the current packet is sent at the sender, we then use the delay controller to perform the packet-level delay control.

3.5 Single-Hop Delay Requirement in End-to-end Delay Guarantee

In the previous sections, we have introduced how to control the single-hop delay based on feedback control and queuing delay adaptation. As our final goal is to control the end-to-end delay of a flow f based on the requirement D^f_{ref}, in this section, we introduce how to set the single-hop delay requirement D^i_{ref}, which is the function of Constraint (3.3) in Section 3.2. As long as each hop can meet its single-hop delay requirement, the desired end-to-end delay is guaranteed.

3.5.1 Worst-case Assignment

The first assignment scheme for the single-hop delay requirement is to use the worst-case analysis approach from Wang et al. (2009b). Specifically, we perform the assignment for the single-hop delay requirement by calculating the worst-case PRR of each hop and assign the single-hop delay requirement proportionally. The worst-case PRR calculation is explained in detail in Wang et al. (2009b). After obtaining the worst-case PRR for each link along the flow, we break the end-to-end delay requirement into a single-hop delay requirement on each hop as:

$$D^i_{ref} = \frac{1/PRR^w_i}{\sum_{u_i \in f}(1/PRR^w_i)} \times D^f_{ref} \tag{3.16}$$

where PRR_i^w is the worst-case PRR for link (u_{i-1}, u_i) on flow f.

The worst-case assignment considers the worst-case scenario in the network such that a hop with a worse PRR^w is assigned a longer single-hop delay requirement. Those nodes with shorter single-hop delay requirements will wake up more often than those with longer delay requirements. This is a pessimistic and static assignment, because the worst-case scenario does not happen frequently in real networks. By assigning an unnecessarily longer delay requirement to the worse links in the worst-case scenario, the nodes in the data flow may have unbalanced energy consumption.

3.5.2 Assignment for Energy Balancing

In this section, we introduce the second assignment scheme to determine the single-hop delay requirements. We propose to periodically update the delay requirement at each hop in the flow. During the transmission of the packet, each node adds the actual average packet reception ratio information of its own receiving link into the packet. The sink will periodically calculate the desired single-hop delay requirement based on the average packet reception ratio at each hop. The sink then sends out the updated delay assignment after each calculation. All the nodes update their single-hop delay requirement information upon receiving this requirement assignment packet. Since the delay requirements are periodically updated based on the current network conditions, each hop can have a fair delay requirement, such that the duty cycle of each receiving node is tuned to be approximately the same, thus leading to the energy consumption balancing. This procedure incurs a small overhead, as the calculation is performed periodically. The period can be set to relatively long if the network condition does not change dramatically.

3.6 Alleviating Communication Interference with Multi-channels

One important technique to improve the communication quality of wireless sensor network is by reducing interference among different sensor nodes in the network. Multi-channel has long been identified as a key tool to reduce the interference as wireless communication in different channels usually do not interfere each others Wang et al. (2009a), Wang et al. (2010). Here we propose a flow-based channel assignment policy to assign channels to each node to further improve the data transmission quality.

We rely on the worst case link PRR analysis in Wang et al. (2009a) and assign channels to node based on the giving end-to-end delay requirement. We need to find a set of disjoint paths from the source nodes to the destination such that the end-to-end delay of each path is smaller than the giving deadline requirement. The delay of a path is calculated as the sum of the weights of all the links in the end-to-end path. Those paths are used to partition the network to form data flows with bounded communication delays. Therefore, the problem of finding Disjoint Paths with Bounded Delay (DPBD) can be formulated as a constrained optimization problem as follows. Given a directed graph G = (V, E) with k source vertices as $s_1, \ldots, s_k$, a destination vertex t, and a set of edges with various weights, we need to find the maximum number of (k or more) mutually vertex-disjoint (except the sources and destination) paths from $s_i, (1 \leq i \leq k)$ to t. The optimization problem is subject to the constraint that the weight (i.e., delay) of each path needs to be smaller than the deadline W. If the number of vertex-disjoint paths is greater than k, some data flows can have more than one path. If we cannot find a path from a source to the destination, the end-to-end delay of that data flow cannot be guaranteed to be smaller than the deadline in the worst case.

3.6.1 NP-Completeness Proof of DPBD Problem

We now prove that the DPBD problem is NP-Complete by reducing it to a well-known NP-complete problem, the Maximum Length-Bounded Disjoint Paths (MLBDP) problem Garey and Johnson (1979), which is stated as follows. Given a graph G' = (V', E') with specified source s, sink t and positive integers k, $W' \leq \|V\|$, does G' contain k or more mutually vertex-disjoint paths from s to t, none involving more than W' edges?

Theorem 3.1. *The DPBD problem is NP-Complete.*

Proof. We first show that DPBD $\in$ NP. Suppose that the solution to the DPBD problem results in k disjoint paths whose lengths are bounded by W. We can verify this solution with a complexity of $O(kW)$, which is in polynomial time. Therefore, DPBD $\in$ NP.

We now reduce our problem to the MLBDP problem. There are two differences between our DPBD problem and the MLBDP problem. First, the edges (i.e., links) in our graph have various weights while the edges in the MLBDP problem have a uniform weight of 1. Second, we need to find one or more paths from each of the k source nodes to the same destination. However, the MLBDP problem aims to find k or more paths between the same source s and the same destination t. We use two steps to reduce our problem to the MLBDP problem.

In the first step, as the weight of each edge is a rational number, we can always find the greatest common denominator for all the edge weights in the graph, which is denoted as c. Thus, the weight of each edge can be expressed as $I \times 1/c$, where I is an integer. We then replace this edge with a chain composed of I new edges (with a weight of $1/c$) and $I - 1$ new intermediate vertices. As a result, the total weight between the two vertices of the original edge is still $I \times 1/c$ while each edge now has a uniform weight of $1/c$. All the edges in the graph can be replaced in the same way, which leads to a new graph where all the edges have the same weight. In the second step, we first add an auxiliary vertex, denoted as s' to the graph G. We then link s'

to each of the k source vertices with an edge whose weight is the uniform value, as shown in Figure **??**(b). If we can find k disjoint paths from s' to t, each source node will have one path to the destination with bounded delay.

After the two steps, we have transformed our graph to a new graph G' = (V',E') with specified vertices s', t and positive integers k, $W' = W \times c + 1$. The DPBD problem is reduced to a new problem stated as follows. Given the new graph G', does G' contain k mutually vertex-disjoint paths from s' to t, none involving more than W' edges? The new problem is exactly the MLBDP problem. Therefore, the DPBD problem is NP-Complete. □ □

3.6.2 Flow-based multi-channel assignment algorithm

In this section, we propose a search algorithm designed based on well-established heuristics Ronen and Perl (1984) to find the required number of disjoint paths in the new graph G' in two steps.

In the first step, the algorithm adopts the Dijkstra's algorithm to find the shortest path from s' to the destination t in the network. If the length of the shortest path is not bounded by W', it is impossible to find k disjoint weight-bounded paths. In that case, the search algorithm fails. If the length of the shortest path is bounded by W', it is added to the solution set T.

In the second step, based on the shortest path found in the first step, the algorithm iteratively searches for the $k-1$ other length-bounded disjoint paths. Every iteration of the algorithm finds a new path, whose length is bounded by W', and guarantees that all the paths found so far are disjoint. Note that each iteration may modify the paths found in previous iterations to maximize the number of length-bounded paths. Specifically, each iteration works as follows.

Starting from s', the algorithm adopts the Depth-First-Search (DFS) method to search for a new path toward t whose length is bounded by W'. Suppose that the search has reached node n and is looking for the next hop. In order to guarantee

that the path found by DFS is disjoint from the existing paths in T, the algorithm first tries to pick the next-hop node of the new path from the neighbors of n that do not belong to any existing paths (referred to as free neighbors). If such a neighbor is available and the total length of the path after adding this neighbor is still smaller than the bound, the neighbor is picked by DFS as the next hop in the new path.

If such a neighbor is unavailable, the algorithm starts an augmentation procedure called *matching*. The procedure checks if n has any neighbor, which belongs to a path in T, can provide a W' bounded path toward t. For example, suppose i is such a neighbor and i belongs to an existing path P in T. The procedure forms a new path P', which includes the current search path from s' to n, the link between n and i, and the part of path P from i to t. If the length of the new path is bounded by W', P is deleted from the solution set T and P' is added to T. The procedure then uses i's predecessor, p_i, in path P as the current node. After the matching procedure, the algorithm starts DFS again from node p_i.

Since DFS may fail to find the next hop and need to back off, the search may go back to node s'. In that case, if all the neighbors of s' have already been visited, it indicates that the last matching procedure was not successful. The algorithm then adds path P, which was deleted in the last matching procedure, back to T, and then removes the new path P' established in last matching from T. The algorithm then rolls back to continue DFS from node p_n, which is the predecessor of the current node n in the last matching procedure.

The whole algorithm terminates under two conditions. First, if the destination t is reached, the algorithm has successfully found a new disjoint length-bounded path. Second, if the search goes back to s' with no more neighbor to visit and all the matching procedures conducted before have been rolled back, the algorithm fails to find a new disjoint length-bounded path. The number of paths in T is the maximum number of disjoint paths with bounded length that the algorithm can find. The detailed algorithm of finding a new disjoint path in the second step is presented in the pseudo code (Algorithm **??**).

Based on the analysis in Ronen and Perl (1984), the time complexity for DFS to find a new path is $O(W'\|E\|)$. The time complexity of the matching procedure in the algorithm is $O(W'^2\|V\|\|E\|)$. Therefore, the time complexity of finding a new disjoint path with bounded delay is $O(W'^2\|V\|\|E\|)$. The algorithm is currently a centralized procedure but the disjoint path search problem can be solved in a distributed way with slightly worse solution quality Sidhu et al. (1991)Cheng et al. (1990). The distributed algorithm includes two steps. In the first step, the sink node sends out a packet to establish a shortest-path tree rooted at the sink in a distributed way. In the second step, the sink sends out messages to explore more paths to the source. Each node individually checks whether a new path is found and notifies the sink to merge the new path into the solution set if the bound requirement is met. This algorithm continues until the desired number of paths is found. Similarly, our solution can also be extended to run on the sensor nodes in the network in a distributed way. The detailed extension is beyond the scope of this paper. In addition, note that many real-world WSN applications (e.g., Selavo et al. (2007)Jeong et al. (2005)) adopt many-to-one communication for data collection, in which the sink is usually a sensor mote connected to the base station, such as a computer. The base station is commonly used to make centralized decisions for these applications. Therefore, our algorithm can also run on the base station computer periodically or in an on-demand manner with PRR values measured at runtime to handle varying network conditions.

3.7 Hardware Evaluation

In this section, we first validate the design of the two major components, the single hop controller and queuing delay adaptation control, in our dynamic duty cycle control (DutyCon) scheme on several simple hardware testbeds. We then evaluate DutyCon under different systematic experiments on a hardware testbed consists of Telosb motes.

Algorithm 2 Finding One New Disjoint W' Bounded Path

Assume we have a solution set T that contains $l \leq k$ disjoint paths;
$n \Leftarrow s'$; Matching Stack Height $\Leftarrow 0$;
while $n \neq t$ **do**
  Use DFS to find next free neighbor $n+1$ that provides W' length bounded path to t;
  if no free neighbor available **then**
    Find a neighbor i in path $P \subseteq T$, which can provide a W' length bounded path to t; Establish path P' through n and i; Push n into the stack; $T \Leftarrow T - P + P'$;

    $n \Leftarrow p\_i$;
    Continue;
  end if
  if no non-free neighbor available **then**
    $n \Leftarrow p\_n$;
    if $n = s'$ **then**
      if matching stack hight $= 0$ **then**
        Return failure.
      else
        $n \Leftarrow$ stack pop out; $T = T - P' + P$;
      end if
    end if
  end if
end while

Table 3.1: PRR values in different periods.

Packet Number	PRR
0-100	0.5
101-200	0.25
201-300	0.5

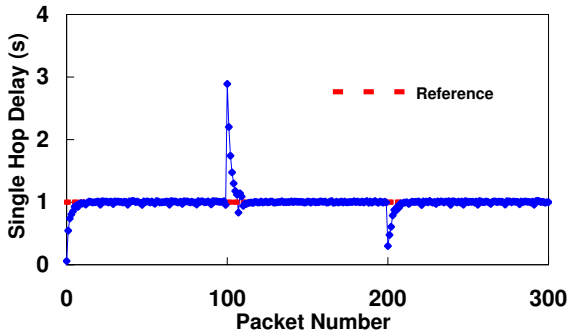

Figure 3.2: Single-hop delay under control when network conditions change at runtime.

3.7.1 DutyCon Components Validation

To verify the design of the single-hop feedback controller on the single-hop delay control performance, we use a pair of Telosb motes. One mote, serving as the sender, generates packets in a uniform distribution with an average rate of 1 packet per 5 seconds. The other mote serves as the receiver and controls its own sleep scheduling based using the single-hop feedback controller. In the first experiment, we set the delay reference of the single-hop transmission to three different values during three different periods in the transmission. The requirements are listed in Table 3.1. The results are shown in Figure 3.2. We can see that when the single-hop delay reference changes, the single-hop controller can approximately control the single-hop delay to the new requirements after a few packets.

In the second experiment, we validate the single-hop controller by emulating the change of network conditions. To emulate the link quality change, we manually set a packet retransmission number for the link at a given period such that packet can only be received correctly after a certain number of retransmission. The retransmission

Table 3.2: Delay references in different periods.

Packet Number	Delay Requirements (s)
0-100	1
101-200	2
201-300	1.5

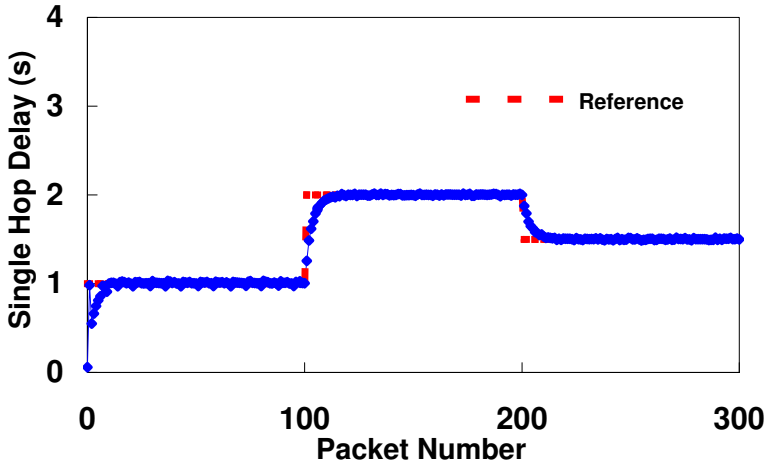

Figure 3.3: Single-hop delay under control when delay requirement changes at runtime.

requirement (PRR) for each period is listed in Table 3.2. The result is shown in Figure 3.3. We can see that despite the two instances at packet 100 and 200 when the network condition changes, delays of all the other packets can be controlled to the delay reference at all time.

We now verify the queuing delay control component using the same single-hop transmission experiment setup. In this experiment, the sender generates 45 packet at

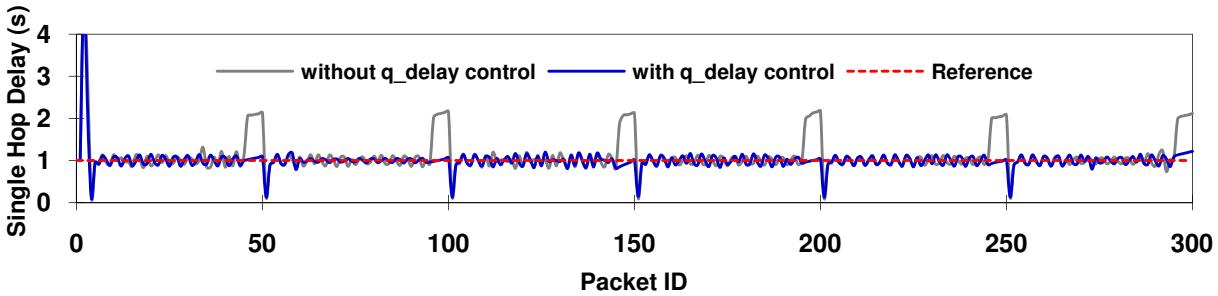

Figure 3.4: Validation result for queueing delay adaptation.

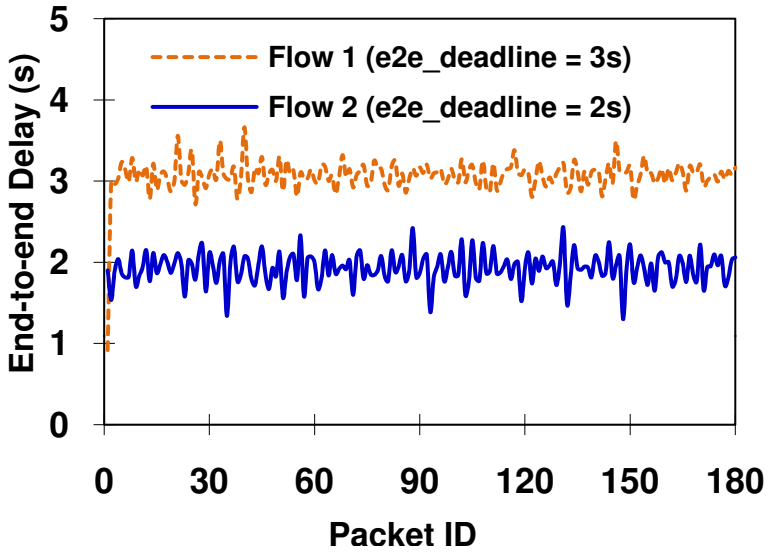

Figure 3.5: Validation result for tree topology.

the rate of 1 packet every 5s and then generate a burst of 5 packets with the rate of 1 packet every 100ms. By following this packet generation model, the burst of packets needs to be put in the queue before transmission. From Figure 3.4 we see that if use single-hop delay controller without the queuing delay adaptation, the single hop delay increases dramatically when there is burst of packets. With queuing delay adaptation, the receiver knows that packets are waiting in the queue at the sender, such that the receiver can adjust its sleep scheduling to adapt to the queuing delay, and the delay can be controlled to the reference even there is burst of packet.

In the last validation experiment, we put all the different components together and use a simple tree-based topology with four Telosb motes. Two motes serve as source nodes and both send packets to a single relay node. The relay node forwards the packet from the two sources to the base station. The end-to-end delay requirements for the two flows are set to 2s and 3s, respectively. The results are shown in Figure 3.5. We can see that the end-to-end delay of both the two flows can be controlled to their own delay requirements, respectively.

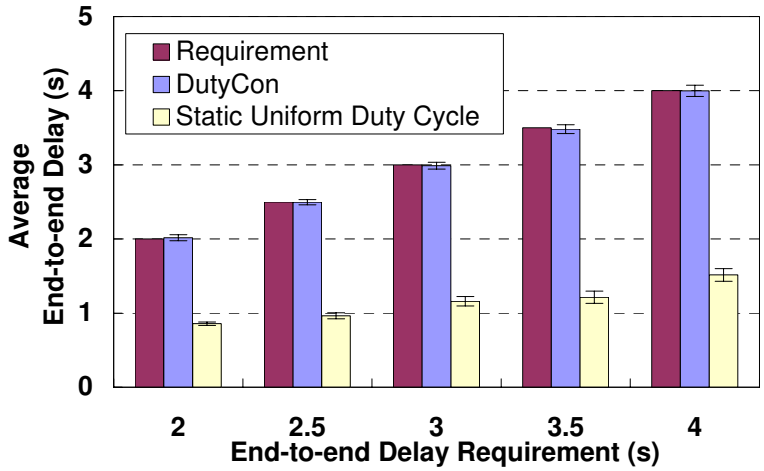

Figure 3.6: End-to-end delay on the testbed under different delay requirements.

3.7.2 Single Flow under Different Deadline Requirements

We now test DutyCon in a scenario of end-to-end communications. The testbed we use consists of 7 Tmote Sky motes, five of which construct a 4-hop end-to-end communication flow. The remaining two motes form a single communication link, periodically transmitting packets and introducing interference to the 4-hop flow. The distance between each two adjacent motes in the 4-hop flow is 5m. The interference link is placed 5m apart from the center node of the flow. The source of the 4-hop end-to-end flow sends out packets, following the uniform distribution at a rate of averagely 1 packet every 3 seconds. The interference link transmits packets at the rate of 1 packet per second. 200 packets are sent by the source in each run of the experiments. We configure the transmitting power of the interference node to be small, so that they will only interfere with the center node of the end-to-end flow.

We evaluate the end-to-end delay control performance of DutyCon on the single flow with different end-to-end delay requirements. We also compare DutyCon with a *Uniform Fixed Duty Cycle* baseline. In this baseline, we set the same fixed duty cycle for every node. The duty cycle we use is the highest duty cycle among all the links from the experiment using DutyCon scheme. The reason we choose this duty cycle is that the local interference is usually unknown a priori. With fixed

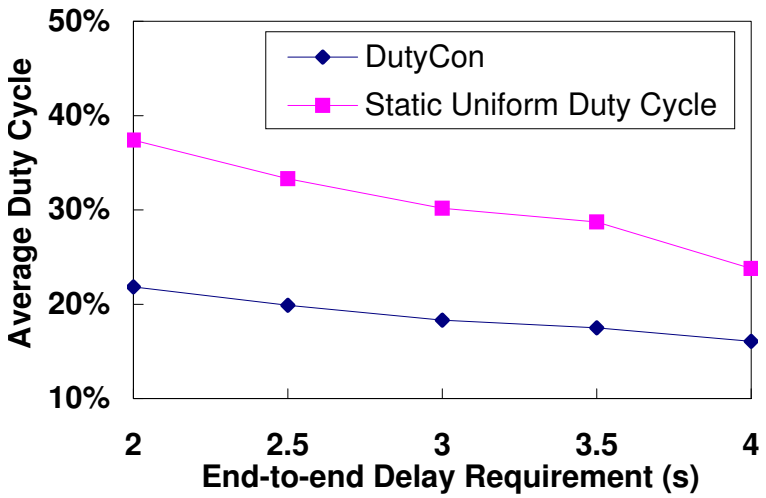

Figure 3.7: Average duty cycle on the testbed under different end-to-end delay requirements.

duty cycle, we want to prepare for the worst-case scenario. Therefore, we use the highest possible duty cycle. From the results in Figure 3.6, we can see that DutyCon can control the average end-to-end delay very close to the delay requirement at all different values. However, using the worst-case duty cycle leads to unnecessarily low end-to-end delays in the *Uniform Fixed Duty Cycle* scheme. Figure 3.7 shows the average duty cycle of all motes in the end-to-end flow except the source node. We can see that the *Uniform Fixed Duty Cycle* scheme has much higher duty cycles (and thus more energy consumption) than DutyCon at all different delay requirements. The reason is that the baseline statically set the duty cycles of all links to the same worst-case value, which leads to an unnecessary energy waste.

3.7.3 Tree Topology under Different Deadline Requirements

In this set of experiments and the experiments in the following two sections, we evaluate the performance of DutyCon in a tree-based topology. The tree-based topology used in the experiments includes 13 telosb motes, constructing 6 converging and intersecting flows in tree topology pattern.

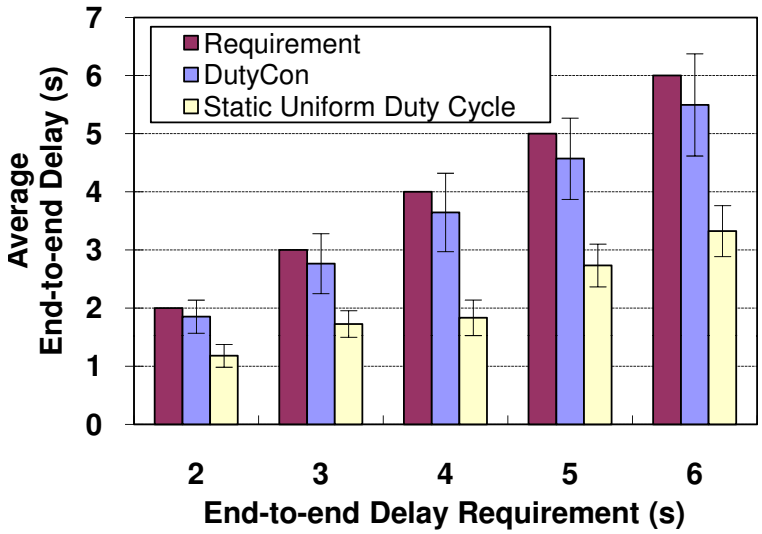

Figure 3.8: End-to-end delay under different delay requirements (tree topology).

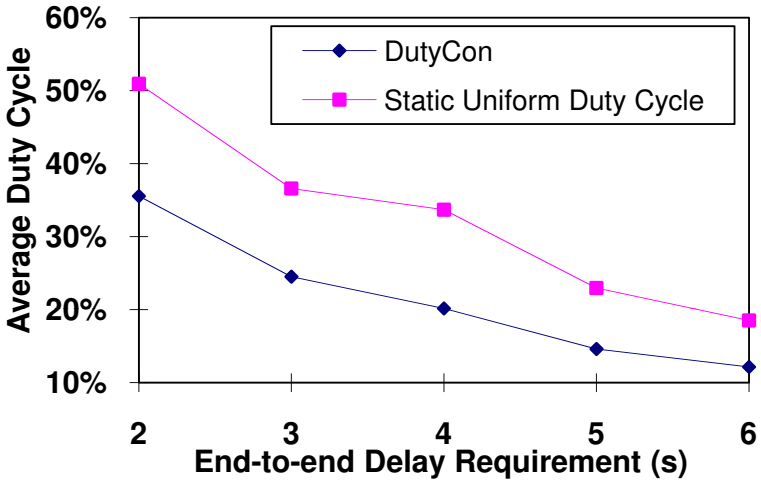

Figure 3.9: Average duty cycle under different end-to-end delay requirements (tree topology).

We first conduct experiment with different end-to-end deadline requirements. We use three flows out of the six flows in the experiments. The end-to-end deadline requirement of each flow is set to the same in each experiment. The source of each flow sends out packet with an average interval of . Figure 3.8 shows the average end-to-end delay now verify the queuing delay control component using the same single-hop transmission experiment setup. In this experiment, the sender generates 45 packet at the rate of 1 packet every 5s and then generate a burst of 5 packets with the rate of 1 packet every 100ms. By following this packet generation model, the burst of packets needs to be put in the queue before transmission. From Figure 3.8 and Figure 3.9 we see that if use single-hop delay controller without the queuing delay adaptation, the single hop delay increases dramatically when there is burst of packets. With queuing delay adaptation, the receiver knows that packets are waiting in the queue at the sender, such that the receiver can adjust its sleep scheduling to adapt to the queuing delay, and the delay can be controlled to the reference even there is burst of packet.

3.7.4 Tree Topology under Different Source Packet Intervals

In this set of experiments, we vary the packet interval at each source node from the interval of 1s to the interval of 6s. The end-to-end delay requirement for each flow is set to 3s. Figure 3.11 shows the average end-to-end delay performance from all the flows. We see that DutyCon can control the average end-to-end delay very close to the delay requirement. This is because it considers the wireless transmission quality dynamically, such that the relay node serving more flows can wake up more often to finish their one hop transmission according to the one-hop delay requirement. Without the dynamic control scheme, the Static Uniform Duty Cycle can only wake up every node with the same sleep scheduling according to the worst-case node. This results in an excessively high duty cycle, which is shown in Figure 3.11, leading to

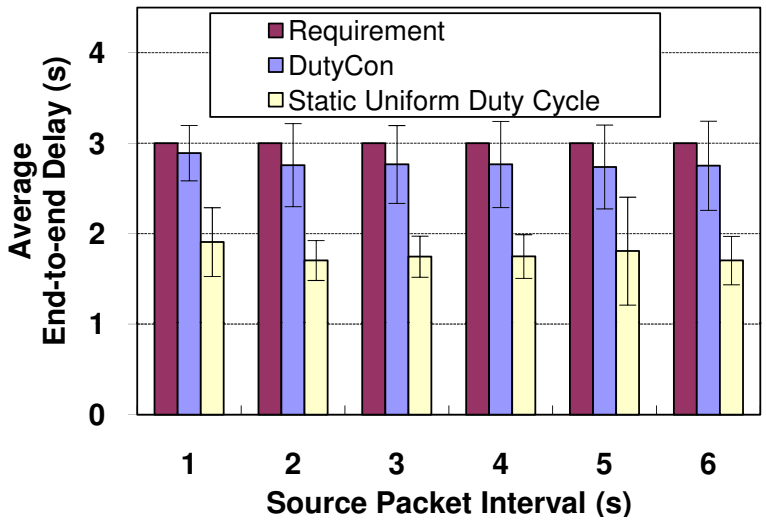

Figure 3.10: End-to-end delay under different source packet intervals (tree topology).

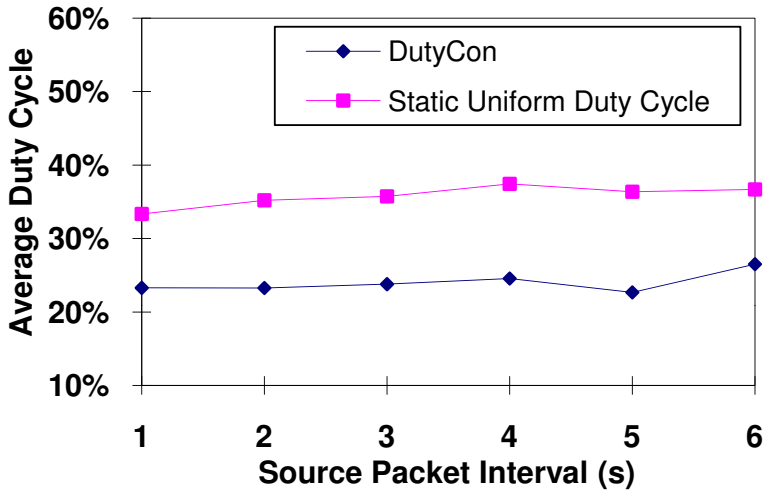

Figure 3.11: Average duty cycle under different source packet intervals (tree topology).

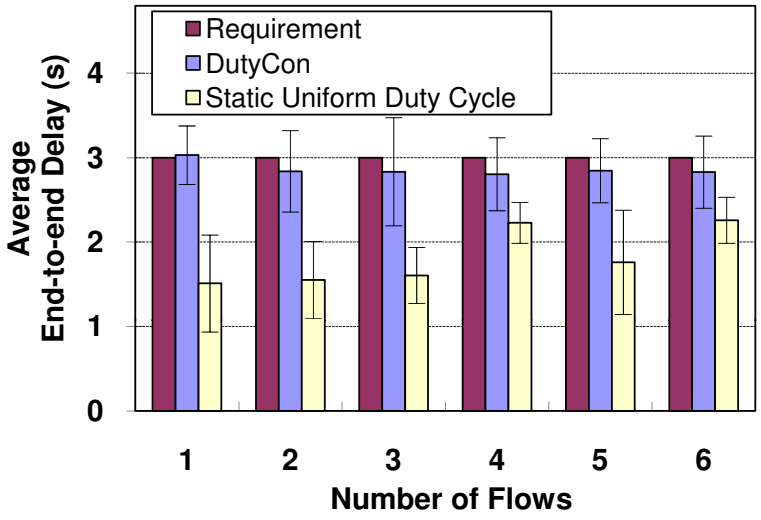

Figure 3.12: End-to-end delay under different number of flows (tree topology).

the unnecessarily low end-to-end transmission delay of every flow and the waste of energy.

3.7.5 Tree Topology with Different Number of Flows

In this set of experiment, we vary the number of flows in our experiment from 1 flow to 6 flows. Each flow has three hops constructed by four nodes. The end-to-end delay requirement of each is set to 3s while the packet interval at each source is set to 3s, too. Figure 3.12 and Figure 3.13 are the average end-to-end delay performance and average duty cycle, respectively. From Figure 3.12 we see that the average end-to-end delay can be controlled close to the requirement by using DutyCon. However, the Static Uniform Duty Cycle scheme uses the worst-case node duty cycle for all nodes in the network, such that the end-to-end delay is unnecessarily low. Both of the two schemes show increasing trend of average duty cycle when more flows are used in the experiment as shown in Figure 3.13. This is because nodes need to wake up more often forward more packets when more flows are sharing one relay node. DutyCon always has a lower duty cycle than the static duty cycle scheme, which leads to substantial energy savings.

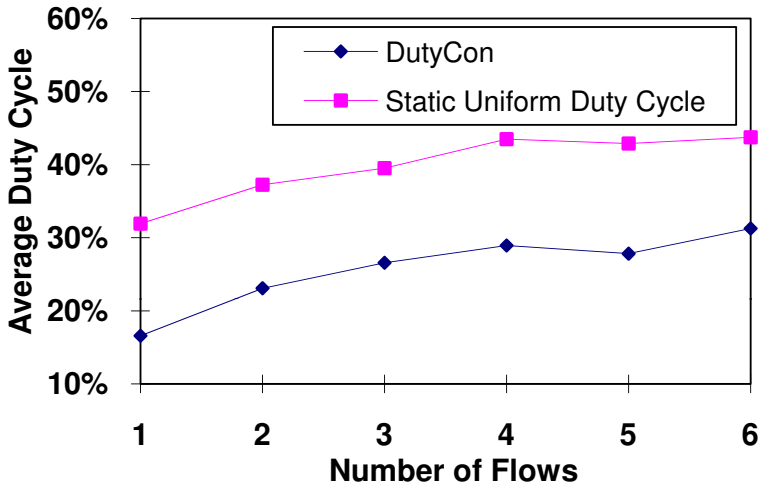

Figure 3.13: Average duty cycle under different number of flows (tree topology).

3.8 Simulation Results

In this section, we present the simulation results in the NS-2 simulator.

3.8.1 Baselines and Simulation Settings

In the following sections, we present the simulation results in the NS-2 simulator. In order to show the efficiency of our design, we compare DutyCon with two recently published baselines: Static Traffic Shaper (STS) and Dynamic Traffic Shaper (DTS) Chipara et al. (2005). Both STS and DTS use traffic shaper to regulate the packet transmitting time at every node such that the end-to-end communication delay can be regulated. The reason we choose these baselines is that STS tries to control the end-to-end communication delay by regulating traffic and DTS tries to regulate packets mainly based on the packet rate, which makes them comparable to DutyCon. Note that it has been demonstrated in Chipara et al. (2005) that STS and DTS outperform SYNC Ye et al. (2002), which features a fixed duty cycle. Therefore, by outperforming STS and DTS, DutyCon also outperforms the baseline protocols used by them.

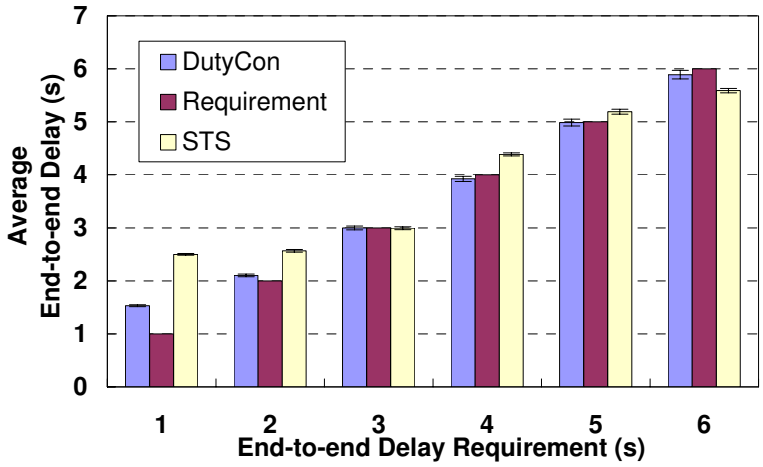

Figure 3.14: End-to-end delay under different delay requirements.

The difference between STS and DTS is that STS enforces a static traffic shaping algorithm according to an end-to-end delay requirement and source data rate, while DTS dynamically shapes the traffic only based on the source data rate. STS decomposes the end-to-end delay requirement and assigns the same delay requirement to each single hop. It also assigns a level to each node based on the distance from the destination. It then regulates the sending time at each node based on the local delay requirements, the node level, and source packet rate. Different from STS, DTS does not enforce the end-to-end delay requirement. It always sets the next packet sending time as the current packet sending time plus the inter-packet time at the source.

Every flow in the following experiments consists of 5 nodes with 100m distance between each hop in a single flow. Flows are randomly deployed in the topology, where each flow has intersection with some other flows. The source of each flow generates packets in a uniform distribution. The average packet rate is one packet every 3 seconds.

3.8.2 Different Delay Requirements

In this set of experiments, we use 3 end-to-end communication flows. We want to test the average end-to-end delays of DutyCon and baselines under different end-to-end delay requirements. Because only the baseline STS enforces an end-to-end delay requirement, we compare the performance of our protocol only to STS.

Figure 3.14 shows the average end-to-end delay performance under different delay requirements. We can see that when the delay requirement is loose (from 3s to 6s), DutyCon can control the average end-to-end delay very close to the desired value. The baseline STS does not have effective control of the average end-to-end delay. When the end-to-end delay requirement is tight (1s and 2s), the average end-to-end delays of both DutyCon and STS are longer than the delay requirement. However, DutyCon performs better than STS in these two cases.

The reason for this result is when the end-to-end delay requirement is loose, DutyCon controls the average delay of each hop to converge to the single-hop delay requirement, such that the average end-to-end delay can be controlled. When the desired end-to-end delay is tight, the delay requirements of each single-hop are too tight such that the controllers of some hops become saturated. When saturation happens, nodes cannot wake up more often. This is because the minimum sleep period of each hop is bounded by the minimum one-hop transmission delay. The saturation of the controller leads to undesired long end-to-end delay. The baseline STS statically estimates the sending time of each packet, based on the source packet rate and the single-hop delay requirement. The randomness of packet generation leads to the inaccurate estimation of the sending time at each hop. Therefore, the end-to-end delay is not well controlled to the desired value. Moreover, for tight end-to-end delay requirements, STS cannot give a good end-to-end delay guarantee. This is because all the nodes of the same level in STS wake up at the same time and try to send packet, which leads to a higher interference degree in the network.

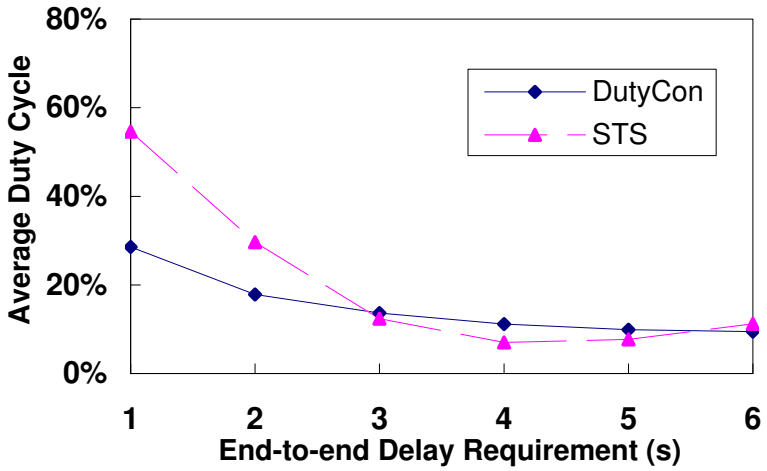

Figure 3.15: Average duty cycle under different end-to-end delay requirements.

Figure 3.15 is the result of the average duty cycle under different end-to-end delay requirements. When the desired delay is tight (1s and 2s), the wakeup time of DutyCon is only half of that of STS. This is because with a tight single-hop delay requirement, the estimation of the sending time by STS is always earlier than the actual packet ready time at the sender, resulting in a long wakeup time on the receiver side. When the delay requirement is 4s and 5s, STS has less duty cycle than DutyCon. However, the two corresponding points in Figure 3.14 show a longer delay than the requirements. When the delay requirement is 6s, STS stays awake for a longer time than DutyCon, which leads to a shorter end-to-end delay in Figure 3.14. The results indicate that when the delay requirements is loose, STS can violate the end-to-end delay requirement; while in other time it consumes unnecessarily high energy for an unnecessarily short delay. Different from STS, DutyCon achieves a good trade off between energy and end-to-end delay by controlling the delay close to the requirement.

3.8.3 Different Flow Numbers

In this section, we evaluate the performance of DutyCon and the two baseline protocols by varying the number of flows in the network. With more flows, there

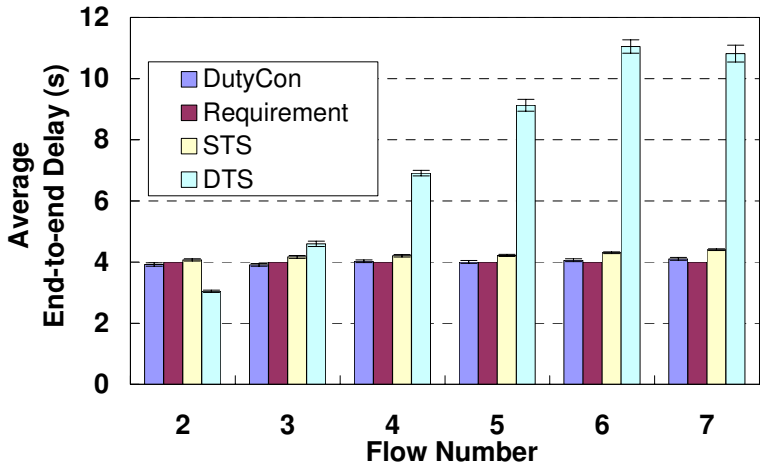

Figure 3.16: End-to-end delay under different number of flows.

will be additional interference in the network. The end-to-end delay requirement of each flow is set to 4 seconds.

Figure 3.16 shows the average end-to-end delay performance. DutyCon can always control the average end-to-end delay close to the requirements. The delay of STS is always higher than the requirement, especially when there are more flows in the network. This is because when the number of flows increases, the number of nodes at the same level increases, leading to a higher degree of interference when they wake up together and try to send packets. DTS has a shorter delay only when the flow number is 2. When the flow number is more than 2, DTS shows a significantly higher delay when compared with DutyCon and STS. The reason is that DTS always estimates the next packet sending time as the previous packet time plus the packet interval at the source. This inaccurate estimation causes the packets to be stacked in the queue, such that all the packets suffer significant queuing delays. Among these three schemes, DutyCon shows the best end-to-end delay as it utilizes the feedback control scheme to dynamically adapt to network environment changes. Furthermore, it features the queuing delay adaptation scheme to adapt the duty cycle to the queuing delay, especially when the number of flow is large. More flows incur more interference in the network, which causes more packets stacked in the queue. However, with

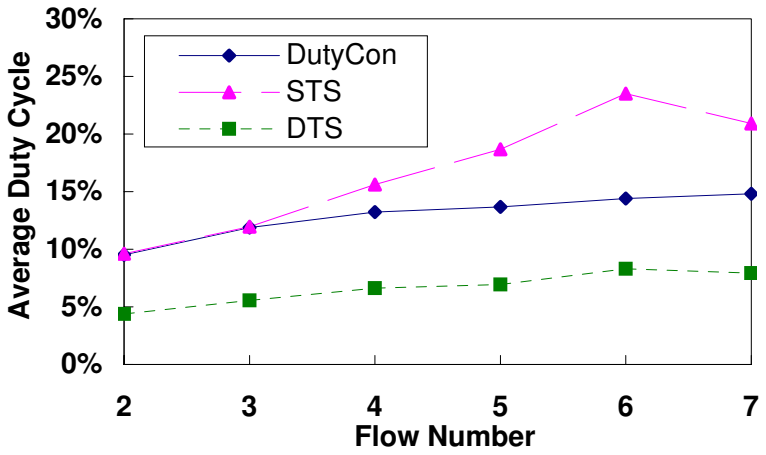

Figure 3.17: Average duty cycle under different number of flows.

the queuing delay adaptation scheme, DutyCon can handle this problem significantly better than the two baselines.

Figure 3.17 shows the average duty cycles of the three protocols with different numbers of flows in the network. DTS has a lower duty cycle in all the situations. This is because the estimation of the next packet sending time is always later than the packet ready time in most cases. Most packets are stacked in the queue, so that when the receiver wakes up, there is always a packet ready for sending at the sender. This leads to a short wakeup time at the receiver. STS shows a higher duty cycle when there are more flows in the network. The higher degree of interference caused by additional flows leads to the longer waking up time in STS. DutyCon has a moderate duty cycle compared with the two baselines.

3.8.4 Benefit of Energy Balancing

In this section, we evaluate the network lifetime and energy balancing on each node under three single-hop delay assignment schemes. The first one is *Worst-case Assignment*, which estimates the worst-case PRR, as introduced in Section 3.5. The second one is the *Energy Balancing* scheme proposed in Section 3.5. In this scheme, we apply the energy consumption balancing scheme to update the delay requirements

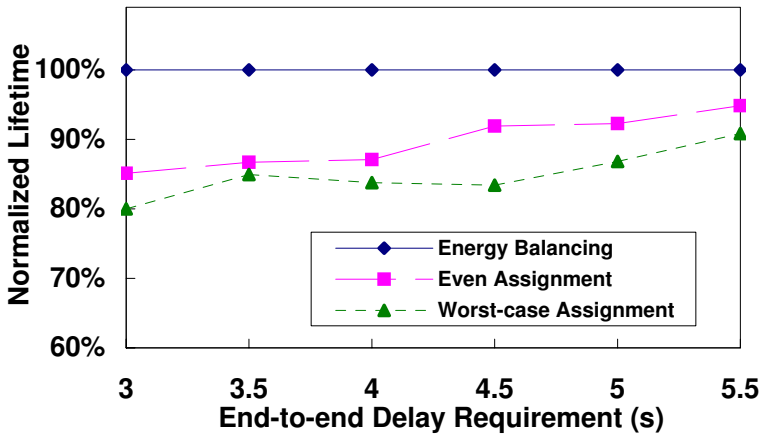

Figure 3.18: Average network lifetime normalized to the maximum under the three single-hop delay requirement assignment schemes.

of each hop periodically, at a rate of every 500 seconds. The third one is called *Even Assignment*, where we assign the delay requirements evenly to each hop of the flow.

Eleven nodes are used to construct a 10-hop single flow in this experiment. Two additional pairs of nodes, forming two single-hop communication links, are placed close to node 5 to introduce interferences to the single flow. The interference signal can be received by nodes 4, 5, and 6. The average packet interval at the source of the end-to-end flow is set to 5 seconds and the packet interval at the interference pairs is set to 1 second. We repeat this end-to-end transmission experiment with different end-to-end delay requirements. The experiment runs until one of the nodes in the flow exhausts all of its initial energy. The entire simulation time of each experiment is considered as the life time of the single-flow network.

Figure 3.18 shows the average network lifetime under each scheme. The values are normalized to the longest lifetime of all the three schemes. The experiment is conducted repeatedly with different delay requirements. The *Energy Balancing* scheme always has the longest network lifetime for all the different delay requirements. When the end-to-end delay requirement is 3 seconds, the *Worst-case Assignment* only achieves an 80% lifetime, normalized to the lifetime of the *Energy Balancing* scheme.

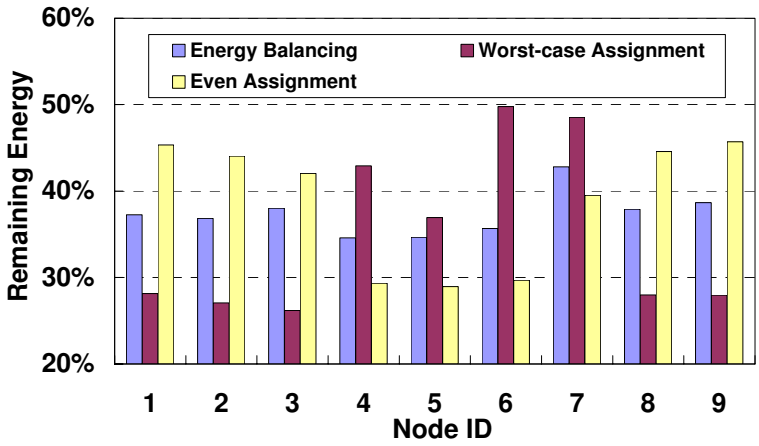

Figure 3.19: Remaining energy of each node under the three single-hop delay requirement assignment schemes.

The results demonstrate that the single-hop delay assignment scheme is critical to the lifetime of the network when employing DutyCon.

We then examine the distribution of the remaining energy among all the nodes. In this experiment, the total number of packets that the end-to-end flow needs to transmit is fixed. Figure 3.19 shows the remaining energy of each node in the single flow after the completion of all the transmissions. When using *Even Assignment*, the remaining energy of nodes 4, 5, and 6 is significantly lower than that of the other nodes. This is because the *Even Assignment* scheme does not consider the different network conditions in different areas of the network. With the same single-hop delay requirement, the interference around nodes 4, 5, and 6 makes the three nodes wake up more often, leading to additional energy consumption. There is a 15% difference in the remaining energy between the most and the least remaining energy from all the nodes. The second scheme, *Worst-case Assignment*, tends to pessimistically assign an unnecessarily long delay requirement to the the hops that are estimated offline to have more interference. Therefore, nodes 4, 5, 6, and 7 have a lower duty cycle than all other nodes, which leads to their lower energy consumption. The difference between the most and least remaining energy under *Worst-case Assignment* is approximately 25% in this experiment. The *Energy Balancing* scheme has the best energy balancing

by periodically updating the delay requirements for each hop, based on the current network conditions. As a result, the difference between the most and least remaining energy is only approximately 8% under *Energy Balancing*.

Chapter 4

Conclusions

In this work, we have applied the frameworks of flocking and PdE to analyze the stability of power grid with a communication network. The similarity between the power grid and the bird flocking, as well as the motivation of using PdE to exploit the network structure is explained. We have considered both cases of negligible and non-negligible communication delays and obtained conditions for the power grid stability in terms of the power/communication network topologies and the system parameters. An algorithm for designing the communication network topology has been proposed, which has been demonstrated by numerical simulations and compared with existing optimization solver, as well as an SDP variation of the problem.

Wireless sensor networks is adopted as the communication infrastructure for the power grid in this work. To effectively utilize WSN in this scenario, we have proposed DutyCon, a dynamic duty cycle control approach that decomposes the end-to-end delay guarantee problem into a set of single-hop delay guarantee problems along each data flow in the network. We then formulate the single-hop delay guarantee problem as a dynamic feedback control problem. DutyCon is designed rigorously based on well-established feedback control theory for analytic assurance of delay control accuracy and system stability. DutyCon also features a queuing delay adaptation scheme that adapts the node duty cycle to unpredictable packet rates,

as well as a novel energy balancing approach that extends the network lifetime by dynamically adjusting the delay requirements allocated to each hop in a data flow. Both empirical results on a hardware testbed and extensive simulations demonstrate that DutyCon can effectively achieve the desired tradeoff between end-to-end delay and energy conservation. Our simulation results also show that DutyCon outperforms two baseline sleep scheduling protocols by having more energy savings while meeting the end-to-end delay requirements.

Bibliography

Abdelzaher, T. F., Stankovic, J. A., Lu, C., Zhang, R., and Lu, Y. (2003). Feedback performance control in software services. *IEEE Control Systems Magazine*, 23:2003. 39

Aleksandrov, A. Y., Hu, G. D., and Zhabko, A. P. (2014). Delay-independent stability conditions for some classes of nonlinear systems. *IEEE Transactions on Automatic Control*, 59(8). 22

Caccamo, M., Zhang, L. Y., and Sha, L. (2002). An implicit prioritized access protocol for wireless sensor networks. In *RTSS*. 38

Cerpa, A., Wong, J. L., Potkonjak, M., and Estrin, D. (2005). Temporal properties of low power wireless links: modeling and implications on multi-hop routing. In *MobiHoc*. 5

Cheng, C., Kumar, S. P. R., and Garcia-Luna-Aceves, J. J. (1990). A distributed algorithm for finding k disjoint paths of minimum total length. In *Allerton '90: Proceedings of the Annual Allerton Conference on Communication, Control, and Computing*. 57

Chipara, O., Lu, C., and Roman, G.-C. (2005). Efficient power management based on application timing semantics for wireless sensor networks. In *ICDCS*. 68

Chuang, F. R. K. (1997). *Spectral Graph Theory*. American Mathematical Society. 11

Crank, J. (1975). *The Mathematics of Diffusion*. Oxford University Press. 17

Currie, J. and Wilson, D. (2012). Opti: Lowering the barrier between open source optimizers and the industrial matlab user. In *Foundations of Computer-Aided Process Operations*. 24, 32

Cvtkovic, M. and Ilich, M. (2011). Pmu based transient stabilization using facts. In *Proc. of IEEE Power Systems Conference and Exposition (PSCE)*. 10

Devane, E. and Lestas, I. (2015). Delay-independent asymptotic stability in monotone systems. *IEEE Transactions on Automatic Control*, PP(99). 22

Ferrari-Trecate, G., Buffa, A., and Gati, M. (2014). Analysis of coordination in multi-agent systems through partial difference equations. *IEEE Trans. Automatic Control*, 51. 4, 13, 14, 17, 18, 22

Franklin, G. F., Workman, M. L., and Powell, D. (1997). *Digital Control of Dynamic Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. 45

Garey, M. R. and Johnson, D. S. (1979). *Computer and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman. 54

Ghosh, A. and Boyd, S. (2006). Growing well-connected graphs. In *Proc. of IEEE Conf. Decision and Control (CDC)*. 27

Grant, M. and Boyd, S. (2014). Cvx: Matlab software for disciplined convex programming, version 2.1. 34

Gu, K., Hkaritonov, V. L., and Chen, J. (2003). *Stability of Time-Delay Systems*. Birkhäuser. 20, 21, 22

Gu, Y., He, T., Lin, M., and Xu, J. (2009). Spatiotemporal delay control for low-duty-cycle sensor networks. In *RTSS*. 39

Gungor, V. C., Lu, B., and Hancke, G. P. (2010). Opportunities and challenges of wireless sensor networks in smart grid. *IEEE Transactions on Industrial Electronics*, 57(10):3557–3564. 4

Ha, R. W., Ho, P.-H., and Shen, X. S. (2006). Cross-layer application-specific wireless sensor network design with single-channel csma mac over sense-sleep trees. *Computer Communications*, 29:3425–3444. 38

He, T., Stankovic, J., Lu, C., and Abdelzaher, T. (2003). SPEED: A stateless protocol for real-time communication in sensor networks. In *ICDCS*. 38

Jeong, J., Culler, D. E., and Oh, J.-H. (2005). Empirical analysis of transmission power control algorithms for wireless sensor networks. Technical Report UCB/EECS-2005-16, EECS Department, University of California, Berkeley. 57

Karenos, K. and Kalogeraki, V. (2006). Real-time traffic management in sensor networks. In *RTSS*. 38

Kundur, P. (1994). *Power System Stability and Control*. McGraw-Hill. 1, 6

Kundur, P. (2008). *Graph implementations for nonsmooth convex program*. Springer-Verlag Limited. 34

Le, H. K., Henriksson, D., and Abdelzaher, T. (2007). A control theory approach to throughput optimization in multi-channel collection sensor networks. In *IPSN*. 39

Li, H. and Han, Z. (2011). Synchronization of power networks without and with communication infrastructures. In *Proc. of IEEE Conference on Smart Grid Communications*. 31

Lin, S., Zhang, J., Zhou, G., Gu, L., Stankovic, J. A., and He, T. (2006). Atpc: adaptive transmission power control for wireless sensor networks. In *SenSys*. 39

Liu, Y. (2012). Wireless sensor network applications in smart grid: Recent trends and challenges. *International Journal of Distributed Sensor Networks*, 2012. 5

Lu, C., Blum, B. M., Abdelzaher, T. F., Stankovic, J. A., and He, T. (2002). RAP: A real-time communication architecture for large-scale wireless sensor networks. In *RTAS*. 38

Lu, G., Krishnamachari, B., and Raghavendra, C. S. (2004). An adaptive energy-efficient and low-latency mac for data gathering in wireless sensor networks. In *IPDPS*. 38

Lyapunov, A. (1992). *General Problem of the Stability of Motion*. CRC Press. 21

Maimour, M. (2008). Maximally radio-disjoint multipath routing for wireless multimedia sensor networks. In *WMuNep*. 40

Merlin, C. and Heinzelman, W. (2008). Duty cycle control for low-power-listening mac protocols. In *MASS*. 39

Min, Y., Yang, L. T., Wang, F., and Wang, W. (2008). Dynamic sleeping algorithm based on ahp for wireless sensor networks. *Future Generation Communication and Networking*, 2:387–392. 39

Ning, X. and Cassandras, C. (2008). Optimal dynamic sleep time control in wireless sensor networks. In *CDC*. 39

Phadke, A. G. and Thorp, J. S. (2008). *Synchronized Phasor Measurements and Their Applications*. Springer. 1

Quera, V., Beltrana, F., and Dolado, R. (2010). Flocking behaviour: Agent-based simulation and hierarchical leadership. *Journal of Artificial Societies and Social Simulation*, 13. 2

Ronen, D. and Perl, Y. (1984). Heuristics for finding a maximum number of disjoint bounded paths. *Networks*, 14:531–544. 55, 57

Selavo, L., Wood, A. D., Cao, Q., Sookoor, T., Liu, H., Srinivasan, A., Wu, Y., Kang, W., Stankovic, J. A., Young, D., and Porter, J. (2007). Luster: wireless sensor network for environmental research. In *SenSys*. 57

Sidhu, D., Nair, R., and Abdallah, S. (1991). Finding disjoint paths in networks. *SIGCOMM Comput. Commun. Rev.*, 21(4):43–51. 57

Thorp, J. S., Seyler, C. E., and Phadke, A. G. (1998). Electromechanical wave propagation in large electric power systems. *IEEE Trans. Circuits and Systems–I: Fundamental, Theory and Applications*, 45. 2, 7

Tuna, G., Gungor, V. C., and Gulez, K. (2013). Wireless sensor networks for smart grid applications: A case study on link reliability and node lifetime evaluations in power distribution systems. *International Journal of Distributed Sensor Networks*, 2013. 5, 37

van Dam, T. and Langendoen, K. (2003). An adaptive energy-efficient mac protocol for wireless sensor networks. In *SenSys*. 38

Wang, X., Wang, X., Fu, X., Xing, G., and Jha, N. (2009a). Flow-based real-time communication in multi-channel wireless sensor networks. In *Wireless Sensor Networks: 6th European Conference*. 53

Wang, X., Wang, X., Xing, F., Xing, G., and Jha, N. (2009b). Flow-based real-time communication in multi-channel wireless sensor networks. In *EWSN*. 40, 51

Wang, X., Wang, X., Xing, G., and Yao, Y. (2010). Exploiting overlapping channels for minimum power configuration in real-time sensor networks. In *Wireless Sensor Networks: 7th European Conference,*. 53

Woo, A., Tong, T., and Culler, D. (2003). Taming the underlying challenges of reliable multihop routing in sensor networks. In *SenSys*. 42, 43, 46

Ye, W., Heidemann, J., and Estrin, D. (2002). An energy-efficient mac protocol for wireless sensor networks. In *IEEE Infocom*. 38, 68

Zhang, Y., Li, X., Zhang, S., and Zhen, Y. (2012). Wireless sensor network in smart grid: Applications and issue. In *Information and Communication Technologies (WICT), 2012 World Congress on*, pages 1204–1208. 5

Zhao, J. and Govindan, R. (2003). Understanding packet delivery performance in dense wireless sensor networks. In *SenSys*. 43

Vita

Xiaodong Wang is the Ph.D candidate in the Department of Electrical Engineering and Computer Science at the University of Tennessee, Knoxville (UTK). He first joined UTK in 2007 and received his Master degree in Computer Engineering from UTK in 2009. He is the recipient of the first Min Kao Scholarships of Electrical and Computer Engineering Department at UTK. He also received ESPN fellowship from College of Engineering at UTK. Xiaodong Wang received his B.S. degree in Electrical Engineering from Shanghai Jiaotong University, China, in 2006. Currently, he is working as a software engineer in F5 Networks Inc. in San Jose, California.