Optimal Sampling Rate Assignment with Dynamic Route Selection for Real-Time Wireless Sensor Networks

Weihuan Shu

Master of Science

School of Computer Science

McGill University

Montréal, Québec

2008-07-01

A dissertation submitted in partial fulfillment of
the requirements for the degree of Master of Science

DEDICATION

To my parents Shen'an Guo and Zhouling Shu

for their love and tolerance

and

To my beloved Yueran

for her warmth and support

ACKNOWLEDGEMENTS

I would like to thank my supervisor Professor Xue Liu for his guidance, support, and encouragement over the last two years. With his enthusiasm, his inspiration, and his great efforts to explain things clearly and simply, he guided me to the way of research and showed me it is not boring but interesting. I would like to also extend my gratitude to Professor Tim Merrett, who served as my advisor when I was new in this university. He provided me many presentation opportunities to build my presentation skills as well as self-confidence, and his critical comments and encouragements were very precious to me. Finally, thank you to all the graduate students in School of Computer Science, McGill University, with whom I enjoy my study and life in Montréal – a wonderful city.

This thesis includes some content from a to-be-published paper in Proceedings of the 28th IEEE Real-Time Systems Symposium (RTSS). I would to thank Professor Liu again for introducing me to this promising yet interesting topic. I also want to thank the other two co-authors for that paper, Professor Zonghua Gu and Professor Sathish Gopalakrishnan, who gave me a lot of valuable advices during writing the paper.

ABSTRACT

The allocation of computation and communication resources in a manner that optimizes aggregate system performance is a crucial aspect of system management. Wireless sensor network poses new challenges due to the resource constraints and real-time requirements. Existing work has dealt with the real-time sampling rate assignment problem, under single processor case and network case with static routing environment. For wireless sensor networks, in order to achieve better overall network performance, routing should be considered together with the rate assignments of individual flows. In this thesis, we address the problem of optimizing sampling rates with dynamic route selection for wireless sensor networks. We model the problem as a constrained optimization problem and solve it under the Network Utility Maximization framework. Based on the primal-dual method and dual decomposition technique, we design a distributed algorithm that achieves the optimal global network utility considering both dynamic route decision and rate assignment. Extensive simulations have been conducted to demonstrate the efficiency and efficacy of our proposed solutions.

ABRÉGÉ

L'attribution de calcul et de la communication ressources d'une manière qui optimise les performances du système global est un aspect crucial de la gestion du système. Réseau de capteurs sans fil pose de nouveaux défis en raison de la pénurie de ressources et en temps réel. Travaux existants a traite distribution temps-reel problème de taux d'échantillonnage, dans un seul processeur cas et réseau cas de routage environment statique. Pour les réseaux de capteurs sans fil, afin de parvenir à une meilleure performance globale du réseau, le routage devrait tre examiné en même temps que la distribution de taux des flux individuels. Dans cet article, nous abordons le problème de l'optimisation des taux d'échantillonnage avec route sélection dynamique pour réseaux de capteurs sans fil. Nous modelisons le probleme comme un problème d'optimisation et le résolvons dans le cadre de l'utilite de reseau maximisation. Sur la base de la méthode primal-dual et la dual décomposition technique, nous concevons un algorithme distribué qui atteint le meilleur l'utilite de reseau globale au vu de route décision dynamique et le taux distribution. Des simulations ont été réalisées pour démontrer l'efficience et l'efficacité de nos solutions proposées.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Introduction

1.1 Wireless Sensor Network

A Wireless Sensor Network (WSN) is a wireless network consisting of spatially distributed autonomous devices using sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, at different locations. Figure 1–1 shows a typical WSN architecture, where each sensor nodes collects some information and send it to the terminal in a multi-hop way. The development of WSNs was originally motivated by military applications such as battlefield surveillance. However, WSNs are now used in many civilian application areas, including environment and habitat monitoring, healthcare applications, home automation, and traffic control. The WSNs are so versatile that they are expected to be the basic building blocks of future pervasive computing environments.

In addition to one or more sensors, each node in a sensor network is typically equipped with a radio transceiver or other wireless communication device, a small microcontroller, and an energy source, usually a battery. The sizes of sensor nodes are usually limited by applications, e.g., for healthcare applications, the sensor nodes should be small enough to be carried easily. A typical WSN application involves from tens to thousands of sensor nodes, therefore the cost is also an important concern. Size and cost constraints on sensor nodes result in corresponding constraints on resources such as energy, memory, computational speed and bandwidth.

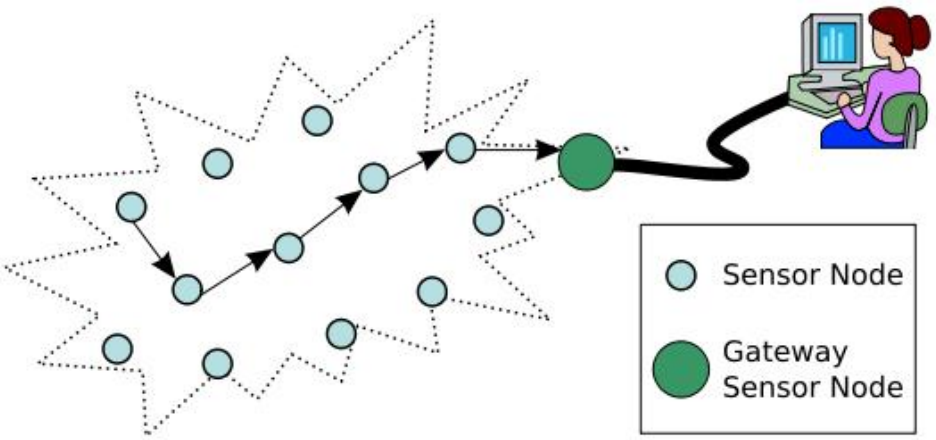


Figure 1–1: Typical Multi-hop Wireless Sensor Network Architecture

1.2 Real-Time System

A Real-Time System is a computing system that must react within precise time constraints to events in the environment. As a consequence, the correct behavior of these systems depends not only on the value of the computation but also on the time at which the results are produced. A reaction that occurs too late could be useless or even dangerous. Today, real-time computing plays a crucial role in our lives, since an increasing number of complex systems rely, in part or completely, on computer control.

Real-time computing is sometimes misunderstood to be high-performance computing, but this is not always the case. For example, a massive supercomputer executing a scientific simulation may offer impressive performance, yet it is not executing a real-time computation. Conversely, a real-time system does not have to be a high-performance computing system, e.g., once the hardware and software for an Anti-lock Braking System (ABS) has been designed to meet its required deadlines, no further performance gains are necessary.

1.3 Real-Time Wireless Sensor Network

A Real-Time Wireless Sensor Network (RTWSN) is a wireless sensor network that can make real-time guarantees. RTWSNs are expected to carry out various applications such as remote control or video/audio monitoring in ad hoc environments. In a RTWSN, the sampling rate is a very important parameter, as it is closely related to the Quality of Service (QoS) of applications. For example, in a RTWSN used for video surveillance, the sampling rate refers to how many frames a video source captures and sends out to the monitoring center per second. For most applications, the higher the sampling rate is, the better the QoS becomes. However, typical WSNs usually face many practical constraints, such as computation limit of the sensors, bandwidth of the routers and delay of the network, which restrict the achievable sampling rates. How to allocate system resources in a way to maximize the aggregate performance of the network subject to these constraints is an important research topic for RTWSN. In this paper, we address the problem of real-time sampling rate allocation and dynamic route selection, with the goal of optimizing the global network performance while maintaining real-time schedulability in RTWSN.

1.4 Organization of the Thesis

The rest of this paper is organized as follows. In Chapter 2, we introduce the background and related work of the problem that we study. Chapter 3 introduces the multi-hop RTWSN architecture employed in this paper. In Chapter 4, we give the real-time schedulability analysis for the system, and show how to improve the utilization of routers comparing to the data transmission scheme used in previous work. The improvement on utilization of individual routers increases network throughput, and this in turn leads to better network utility. In Chapter 5, we model the optimal sampling rate assignment with

dynamic route selection problem as a nonlinear optimization problem. The centralized algorithm and the distributed algorithm are both presented in Chapter 6. In Chapter 7, we conduct a bunch of simulations, show the results and analyze them: we first show that the distributed algorithm is efficient by analyzing the simulation results on convergence, and then prove that our proposed data transmission scheme is better by comparing it with the original scheme in [25]; we also study how the step size of updating and the convergence threshold affect the convergence, and compare the distributed and centralized algorithms in terms of control traffic and scalability. Finally, Chapter 8 presents conclusions and future work.

CHAPTER 2
Related Work

2.1 Resource Allocation in Computer Networks

Resource allocation has been an active research area for computing systems [22], [34], [3]. Most of them do not take real-time requirements into consideration and hence cannot be directly applied to real-time systems. Kelly and Low et al. first studied the problem of resource allocation for congestion control in computer networks [21], [20], [26]. These papers focused on the the optimization problem given link capacity constraints, and they formed the foundation of Network Utility Maximization (NUM), but again the real-time constraints were not considered. Later, researchers extended the work to resource allocation in wireless networks in general [9], [7] and in WSNs in particular [13].

Finding the optimal task execution rates subject to the schedulability constraints was first studied by Seto et al. and by Sha et al. for analog and digital controllers, respectively [29], [31]. They presented offline optimization techniques based on the Kuhn-Tucker conditions[1] , but the schedulability constraint considered is only for single processor. Rajkumar et al. developed QoS-based Resource Allocation Model (Q-RAM), which is capable of handling multiple quality dimensions [28], but the solution can only be used in

[1] More details about Kuhn-Tucker conditions can be found in [2].

a single constraint case. Lee and Ghosh et al. studied the scenario under multiple constraints [23], [14], but the problem they addressed is an integer programming problem, which is different from the problem discussed in this paper. In [23], the integer programming problem proved to be NP-hard, and several sub-optimal algorithms were proposed. According to [14], Hierarchical Q-RAM is the technique with the best scalability. However, that algorithm requires the division of multiple constraints into independent groups, which is impractical for multi-hop RTWSN. Lately, the work by Chen et al. made it possible to achieve the maximum system utility [8], but it considered discrete task rates, and the employed model as well as the focused problem are different from ours.

2.2 Resource Allocation in RTWSNs and Network Utility Maximization

RTWSN presents new challenges for real-time resource allocation. First, size and cost constraints require an efficient and distributed algorithm that can globally optimize the resource allocation. Second, since routes in a RTWSN may intersect with each other at routers, sampling rate optimization for real-time flows must take into consideration the traffic contention at each router. Liu et al. first transformed the real-time sampling rate assignment problem in a WSN to a constrained optimization problem [25], which explicitly captures the real-time requirements of the WSN as optimization constraints. They also proposed a distributed algorithm based on the Internet pricing schemes [26]. However, this work assumed that packet routing decision is made independent of rate selection, and routes stay unchanged during the process of sampling rate optimization. As we will show later in Section 7.1, this assumption of static route selection may limit the global network performance seriously, which is also referred to as *network utility*.

The resource allocation problem in RTWSNs is a complex problem, which involves more than one layer in the network protocol stack. For example, to achieve the optimal QoS or network utility in the network, multiple layers have to be taken into accounted: how to set the sampling rates is a problem belonging to the physical layer; the Medium Access Control (MAC) layer is responsible for efficiently dealing with the contention or transmission failure; how to set up the routing in order provide the best service is a concern of network layer (routing layer); etc. More importantly, to achieve a global optimum, multiple layers have to be considered together. As an example, we will show later that, to consider the physical layer and the routing layer separately is not enough to achieve the optimal network utility.

Recently, based on the NUM framework, extensive research has been conducted towards a systematic understanding of "layering" as "optimization decomposition", where the overall communication network is modeled by a generalized NUM problem: each layer corresponds to a decomposed sub-problem, and the interfaces among layers are quantified as functions of optimization variables coordinating the sub-problems [9]. For example, the problem of joint optimization of congestion control and routing has been studied by Chen, Lin, Wang and He et al. in [7], [24], [35], [16], the problem of joint optimization of congestion control and scheduling has been studied by Eryilmaz, Andrews and Marbach et al. in [12], [1], [27], and the problem of joint optimization of routing, scheduling and power control has been studied by Cruz and Xi et al. in [11], [36]. Their approaches showed how a joint optimization problem can be decoupled and separated into different network layers.

In the above mentioned works, Wang et al. interpreted TCP with Active Queue Management (TCP-AQM) as distributed primal-dual algorithms to maximize aggregate utility

over source rates [35]. The flow rates and routing are the optimization variables, where the link capacity constraints are considered. Our work is inspired by this paper, and we manage to introduce the real-time constraints to the original NUM problems, in order to provide real-time services for wireless sensor networks.

2.3 Our Contribution

In this paper, we systematically study the problem of optimal sampling rate assignment together with dynamic route selection for real-time wireless sensor networks. In contrast to the work by Liu et al. [25] where static routing is assumed, we allow *dynamic routing*. In our model, each sensor source has one or more paths leading to its corresponding destination (data sink), but only one path at a time is selected for data transmission. The set of candidate paths between a source and a destination can be chosen offline based on existing routing algorithms for wireless sensor networks such as SPIN [18], GPSR [19], GEAR [37], Rumor Routing [4], SPEED [17] or RPAR [10]. Instead of using the "optimal" route determined by a specific routing algorithm, we keep all the feasible ones as candidate routes according to application requirements and select route to maximize the overall network utility.

We first show that the data transmission scheme employed in [25] is not efficient when the data blocks to be transmitted are relatively large, then propose a new scheme that can enhance the network utility. The new scheme also facilitates schedulability analysis for each router, as well as the implementation of the distributed algorithm. The optimization problem with dynamic route selection is then formulated and transformed into an optimization problem with nonlinear objective function and linear constraints. Finally, a distributed algorithm based on the primal-dual method and dual decomposition technique

will be given for the joint optimization problem. The algorithm is able to find the optimal sampling rates and the optimal routing, while maintaining the real-time schedulability in a dynamic routing environment.

CHAPTER 3
System Architecture

3.1 RICH Architecture

Caccamo et al. first provided real-time support for multi-hop RTWSN [6], where a cellular base station layout is deployed as the backbone for the underlying RTWSN, as shown in Figure 3–1. In this architecture, each base station functions as a router at the center of each cell. The base stations use seven non-overlapping Radio Frequency (RF) bands, and all RF broadcasts are within one-hop. The particular geographical layout makes each base station and its six neighbors transmit with distinct RF bands, and any two base stations sending with the same RF band are at least two cells apart. The inter-cell communication in the wireless sensor network uses a globally synchronized TDMA scheme, where a period is divided into six slots, each corresponding to the data transmission towards one of the six directions. Therefore, the inter-base-station communication is a mixed FDMA-TDMA scheme.

More recently, based on the mixed FDMA-TDMA scheme, Giannecchini et al. provide an online suboptimal approximation algorithm (CoRAl) to dynamically reconfigure sensing rates of RTWSN [15]. CoRAl runs fast but only applies to exponential performance loss function.

Based on the cellular base station backbone layout in [6], Liu et al. proposed the Real-time Independent CHannels (RICH) architecture [25]. RICH architecture employs the

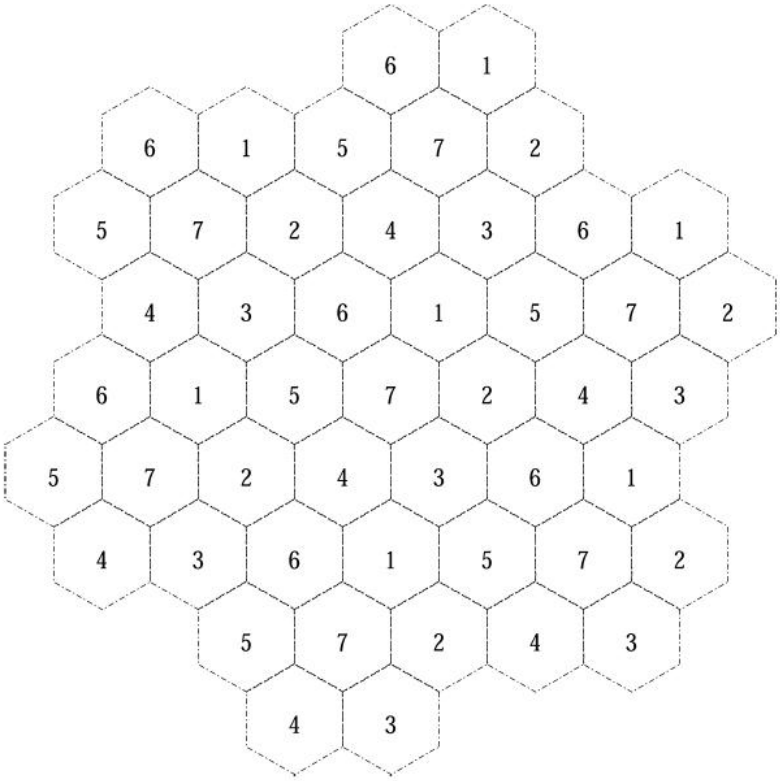


Figure 3–1: The Mixed FDMA-TDMA Base Station Backbone Layout

mixed FDMA and Direct Sequence Spread Spectrum CDMA (DSSS-CDMA)[1] scheme instead of the mixed FDMA-TDMA scheme [6], in order to achieve better flexibility and simpler schedulability analysis. Figure 3–2 shows the internal architecture of the RICH base station. Each RICH base station has seven DSSS-CDMA modulation/demodulation Co-Processing Units (CoPUs), each operates with a distinct DSSS-CDMA Pseudo Noise (PN) sequence at a distinct FDMA RF band. Among which, six of the DSSS-CDMA CoPUs are receivers, and the other one is the only transmitter of the base station.

Moreover, they deployed forty-nine DSSS-CDMA PN sequences and maintained a seven RF band coloring of the cells, denoted as $A1, \ldots, A7, B1, \ldots, B7, C1, \ldots, C7$, $D1 \ldots, D7, E1, \ldots, E7, F1, \ldots, F7, G1, \ldots, G7$ respectively. Figure 3–3 shows this deployment. In a cell labeled XY, the RICH base station transmitter deploys the XYth PN sequence for DSSS-CDMA modulation, and transmits at the Yth RF band. For example, the base station in a cell labeled $G7$ transmits with the $G7$th DSSS-CDMA PN sequence at the 7th RF band. The transmission range of every transmitter in our RICH RTWSN is within one-hop. Each of the six receivers on a RICH base station listens to one of its one-hop neighbors transmission. Take the RICH base station at a cell labeled $A5$ for example, its six receivers listen to the 6, 7, 4, 1, 3, 2th RF band respectively, and demodulate with DSSS-CDMA PN sequence $A6, A7, A4, G1, F3, F2$ respectively.

Under such design, the broadcast of a base station is simultaneously received by its six one-hop neighbors. Due to the employment of DSSS-CDMA technique, transmitting

[1] Nowadays, the term CDMA usually refers to DSSS-CDMA. A brief tutorial of DSSS-CDMA can be found in [25].

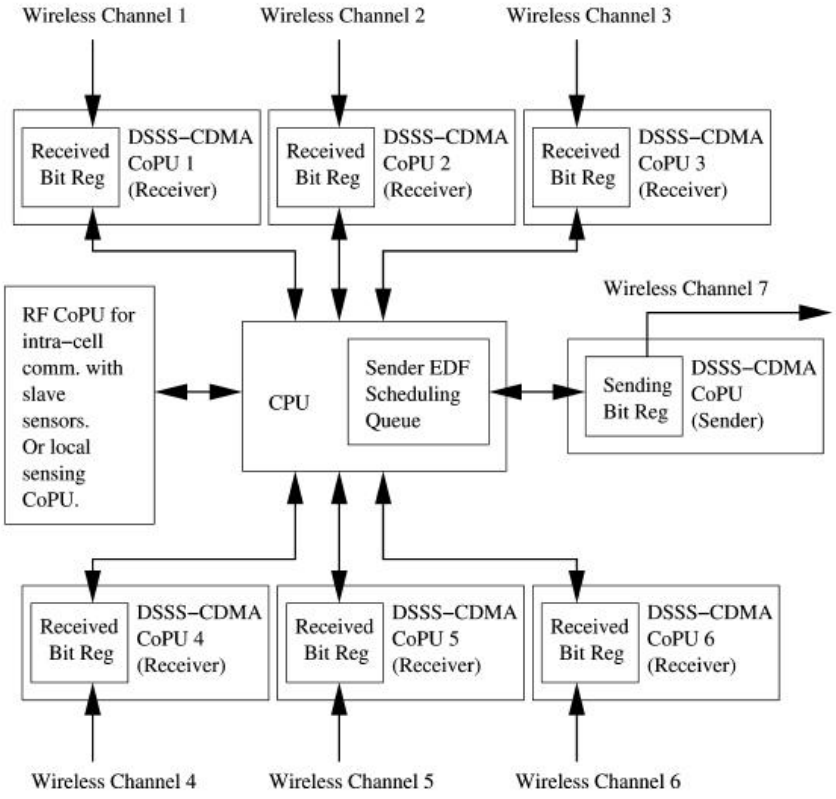


Figure 3–2: The Internal Architecture of a RICH Base Station

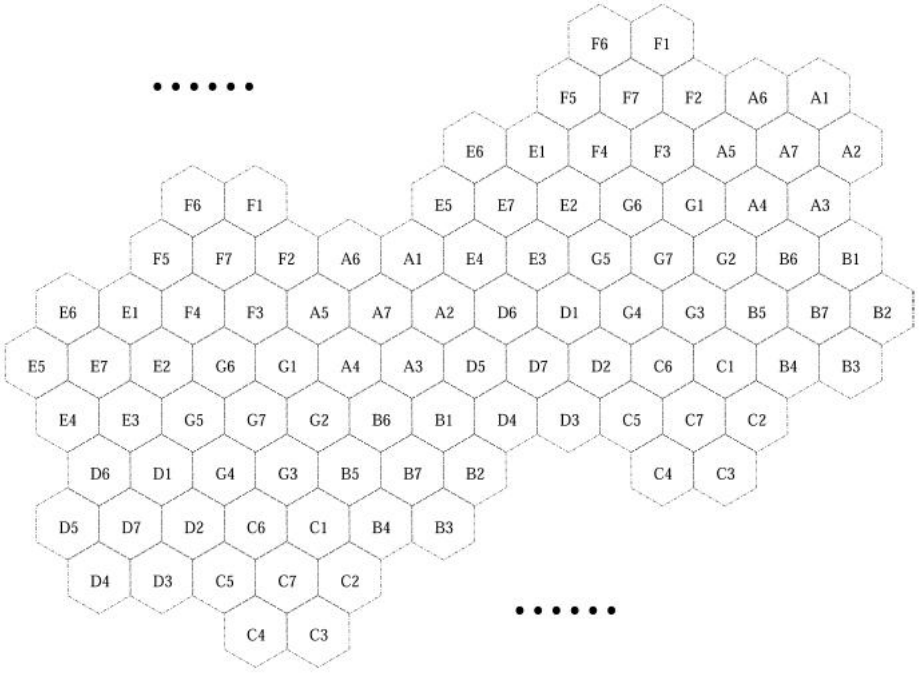


Figure 3–3: The Mixed FDMA-CDMA Base Station Backbone Layout

and receiving can be carried out independently, and there is no synchronization needed between any pair of transmissions. Furthermore, the bandwidths of each DSSS-CDMA CoPU can be distinct, and the scheduling can be independently adjusted to be specific to each base station.

In this paper, we adopt the mixed FDMA-CDMA RICH architecture for RTWSN deployment. Note that under the RICH architecture, there can be multiple wireless sensors (slaves) deployed inside each cell. These sensors usually perform the actual sensing and also communicate with the base station (head sensor) of the corresponding cell at different RF bands that do not interfere with the inter-cell communications. This paper only focuses on the inter-cell communication. Since intra-cell communication is local to the cell, it is not the focus of this paper, and is therefore not addressed explicitly.

3.2 Schedulability Modeling

For a given RICH base station n, the available bandwidths to its six neighboring RICH base stations are $B_n^1, B_n^2, \ldots, B_n^6$ respectively, which may be different from each other due to irregularity of the wireless medium [38]. We set the data transmission bandwidth of base station n to be $B_n = \min_i B_n^i, 1 \leq i \leq 6$, hence the broadcast of n can be reliably received by all its six neighbors. In the rest of this paper, we use *node* to refer to the base station that acts as a router, *source* to refer to the base station that has data originated within the corresponding cell, and *destination* to refer to the last router for data transmission of a particular application. Note that a source is also a router, since it always forwards data collected from the sensors within the cell. We also assume that the data are transmitted to the application terminal by ways other than the mixed FDMA-CDMA inter-cell scheme.

Let $\mathcal{S}$ be the set of sources in the network, and let $\mathcal{S}_n$ be the set of sources for which node n forwards data. Assume a source $s \in \mathcal{S}_n$ has a sampling/reporting rate of f_s, and each report packet has a length of l_s. The corresponding transmission time for a packet from source s on node n is therefore $c_s = l_s/B_n$. As the sampling/reporting rate is fixed, the data transmission for $\mathcal{S}_n$ on node n can be viewed as a periodic task set, with the transmission for an individual packet from $s \in \mathcal{S}_n$ as a job and all transmissions for $s \in \mathcal{S}_n$ as the periodic task.

Existing scheduling algorithms can be used to determine the schedulability, such as Rate Monotonic (RM) and Earliest Deadline First (EDF) scheduling algorithms. In this paper, we choose non-preemptive EDF scheduling algorithm because EDF is simple for analysis and packet transmission is usually non-preemptive. The detailed schedulability analysis is given in the next chapter.

CHAPTER 4
Schedulability Analysis

In this chapter, we discuss the schedulability problem in real-time data transmission. Non-preemptive EDF scheduler is employed to schedule the packets that are received and to be transmitted at each router. The following sections first introduce the EDF scheduling (preemptive and non-preemptive) shortly, and show how to model the non-preemptive EDF schedulability problem for data transmission.

4.1 Preemptive EDF

EDF is a dynamic priority scheduling algorithm that always selects the task with the shortest absolute deadline to execute, in other words, the task with the earliest deadline has the highest priority. From the aspect of a single processor, if all tasks are periodic, preemptive, and have deadlines equal to their periods, EDF is able to achieve 100% processor utilization. Since $f(frequency) \times T(period) = 1$, we can write the schedulability condition as

$$\sum_{\tau \in \mathcal{T}} c_\tau f_\tau \leq 1,$$

where τ is a periodic task in task set $\mathcal{T}$, c_τ is the computation time of task τ and f_τ is the frequency of task τ.

4.2 Non-preemptive EDF

Although preemptive EDF scheduling is optimal, data transmission is a task that cannot be interrupted once it begins, so non-preemptive scheduling must be used for practical applications. A network data packet typically consists of a header section with the sequence number, type and routing information, and a data section with the actual data payload. It is well-known that transmission of a data packet should be atomic and not be interrupted in the middle, since missing any bits from the header section will influence data transmission and missing any bits from the data section will violate data integrity. The schedulability condition for non-preemptive EDF [5] is

$$\sum_{\tau \in \mathcal{T}} c_\tau f_\tau + C_i f_i \leq 1, \text{ for each } i \in \mathcal{T}, \tag{4.1}$$

where $C_i = \max_{\{\tau \in \mathcal{T} \text{ and } \tau \neq i\}}\{c_\tau\}$ is the maximum blocking time for task i.

4.3 EDF Scheduler for Data Transmission

In data transmission scenario, the constraint (4.1) for node n can be transformed into

$$\sum_{s \in \mathcal{S}_n} \frac{l_s}{B_n} f_s + \frac{L_i}{B_n} f_i \leq 1, \text{ for each } i \in \mathcal{S}_n,$$

where l_s is the packet length for source s and $L_i = \max_{\{s \in \mathcal{S}_n \text{ and } s \neq i\}}\{l_s\}$ is the maximum packet length among all the packets that may block the data transmission for source i. Then we have the following schedulability condition for data transmission:

$$\sum_{s \in \mathcal{S}_n} l_s f_s + L_i f_i \leq B_n \text{ , for each } i \in \mathcal{S}_n. \tag{4.2}$$

By taking into account the header of each packet, we have $l_s\,(PacketLength) = h_s\,(HeaderLength) + d_s\,(DataLength)$ for each source s. In [25], the data from each source

are directly encapsulated into packets with different sizes that are application-specific. This scheme is appropriate for small packet sizes, but may adversely affect system performance with large packet sizes due to the blocking time term in inequality (4.2), which is related to the maximum length of packets from other sources. We give an example to illustrate this issue.

Assume that node n with a total bandwidth of $1.92Mbps$ acts as a router for two sources s_1 and s_2, whose parameters are given in Table 4–1.

Table 4–1: Parameters of the Example

	$f\ Hz$	$l\ Mb$
s_1	5	0.2
s_2	10	0.01

In real-time computing, *utilization* is the proportion of the system's resources which is used for computation. For a period task set, the utilization is defined as $\sum_{\tau\in\mathcal{T}} c_\tau f_\tau$. We define *achievable utilization* of a node n to be the maximum utilization of a node while satisfying the schedulability condition, i.e., the maximum value of $\sum_{s\in\mathcal{S}_n} c_s f_s$ (or $\sum_{s\in\mathcal{S}_n} l_s f_s / B_n$) such that constraint (4.2) can be satisfied. We also define the *leftover bandwidth* as the extra bandwidth capacity of node n to transmit more data:

$$\min_{i\in\mathcal{S}_n} B_n - \Big(\sum_{s\in\mathcal{S}_n} l_s f_s + L_i f_i\Big).$$

If node n only forwards packets for s_1, then the schedulability condition (4.2) for source s_1 is $0.2 \times 5 \leq 1.92Mbps$, which is satisfied. The leftover bandwidth of node n is $1.92 - 0.2 \times 5 = 0.92Mbps$ which seems to be sufficient to forward additional data from source s_2. However, it turns out that node n cannot forward data from both s_1 and s_2 simultaneously due to blocking. If node n forwards packets for both s_1 and s_2, then the

schedulability condition (4.2) includes two inequalities, in which $0.2 \times 5 + 0.01 \times 10 + 0.2 \times 10 \leq 1.92 Mbps$ does not hold. Although the packet size of s_2 is very small, and half of the bandwidth of node n has not been utilized, it is still impossible to meet the schedulability condition. In fact, no matter how small the packet size of s_2 is, the term due to blocking time from packets of s_1 is $0.2 \times 10 = 2 Mbps$, which will make the condition (4.2) false. We call this phenomenon *utilization jump*. It greatly limits the achievable utilization of the nodes and therefore limits the overall network performance. This problem is especially severe for video sensor networks, where nodes send high-resolution video frames periodically at high sampling rates, but is less of a concern for other application, where nodes send scalar measurement values such as temperature, pressure, humidity, etc.

In this paper, we propose a solution for this problem by dividing a large data block from a given data source into multiple smaller fixed-size packets with the same deadline as the original data block. Packets with the same deadline have the same priority, and are processed in FIFO order. We call this new scheme *packet transformation*, in analogy with the technique of *period transformation* in prioritized preemptive scheduling [30]. Since the maximum packet length that can block a transmission task is l, the fixed packet length, we can re-write inequality (4.2) as

$$\sum_{s \in \mathcal{S}_n} l\, k_s f_s + l f_i \leq B_n \text{ , for each } i \in \mathcal{S}_n, \tag{4.3}$$

or

$$\sum_{s \in \mathcal{S}_n} k_s f_s + f_i \leq \frac{B_n}{l} \text{ , for each } i \in \mathcal{S}_n, \tag{4.4}$$

where $k_s = \lceil \frac{p_s}{d} \rceil = \lceil \frac{p_s}{l-h} \rceil$ is the number of packets resulting from dividing the data block of source s with size p_s.

Compared to the scheme in [25], *packet transformation* increases the achievable utilization by reducing the utilization jump, since the length of a blocking packet is bounded in the new schedulability condition (4.3). However, it may increase system overhead due to an increased number of packet headers. Therefore, we should choose packet sizes judiciously to strike a balance between the utilization waste due to utilization jump and overhead due to packet headers, as we will show in Section 7.3.

Based on current wireless transmission technology, the data rate of a wireless base station can be high and will become even higher. The possibility of higher data rate raises applications with larger data blocks. High data rate cannot alleviate the utilization jump problem, but it greatly decreases the disadvantage brought by header overhead. Therefore, as we will show later in the evaluation chapter (Section 7.3), packet transformation is able to achieve better network utility by balancing the utilization waste due to utilization jump and overhead due to packet headers.

CHAPTER 5
Mathematical Formulation

In this chapter, we present the formal problem formulation of the optimal joint sampling rate assignment and dynamic route selection problem for real-time wireless sensor networks. Our formulation models the problem as a nonlinear convex optimization problem with linear constraints as follows:

$$\min_{\boldsymbol{f},\, \mathbf{R}\in\mathcal{R}_s} \sum_{s\in\mathcal{S}} U_s(f_s) \tag{5.1}$$

subject to

$$\boldsymbol{f} \leq \boldsymbol{f}^{max} \tag{5.2}$$

$$\boldsymbol{f} \geq \boldsymbol{f}^{min} \tag{5.3}$$

$$\mathbf{A}\boldsymbol{f} \leq \boldsymbol{b}. \tag{5.4}$$

In the above formulation, $\boldsymbol{f}$ and $\mathbf{R}$ are the decision variables representing the flow rates and routes to be decided. $U_s(f_s)$ is a function measuring the utility loss of the real-time flow originated from source s, and $\mathcal{S}$ represents the set of sources in the network. The formulation is discussed in detail in the following sections.

5.1 Network Utility Loss Index

For most applications, performance (QoS) improves with increasing of sampling rates. Ideally, the best performance is achieved with infinite sampling rate, i.e., continuous sampling, which is obviously not achievable in reality. We use the *Utility Loss Index*

(*ULI*) to capture the performance loss using discrete sampling rates compared to the case when using continuous sampling [29]. For control applications, Seto et al. showed that the ULI is in the following general form:

$$U_s(f_s) = \omega_s \alpha_s e^{-\beta_s f_s},$$

where f_s is the sampling rate of source s, and non-negative values ω_s, α_s and β_s are application-specific parameters, which can be determined through curve fitting using measurement data. In this paper, we generalize the form of ULI function to strictly decreasing differentiable convex function with regard to rate f_s. The sum of ULI over all the sources in the network is defined as *network ULI*, which is the objective function of our formulated optimization problem. The network utility maximization can be achieved by minimizing the network ULI.

5.2 Single-path Routing

We first introduce some notations used to model the network and routing. They will be encapsulated in the constraints formulation.

B_n Broadcast bandwidth of node n.

$\boldsymbol{b}$ Broadcast bandwidth vector.

K^s Number of acyclic paths from source s to its destination.

$\mathcal{S}$ The set of sources in the network.

S Number of sources in the network.

$\mathcal{N}$ The set of nodes in the network.

N Number of nodes in the network.

$\mathcal{L}$ The set of directional links in the network.

L Number of directional links in the network.

To help the readers understand our modeling better, we also give a practical example in this section. Consider the wireless sensor network shown in Figure 5–1, where nodes n_1, n_3, n_4, n_{11} and n_{14} are sources (numbered $s_1, \ldots, s_5$ respectively) that send data to their corresponding destinations n_{15}, n_{16}, n_1, n_{13} and n_7 (numbered $d_1, \ldots, d_5$ respectively).

We suppose the following candidate routes between the sources and the corresponding destinations are obtained with an existing routing algorithm:

$$\mathcal{H}^1 = \begin{cases} n_1 \rightarrow n_2 \rightarrow n_5 \rightarrow n_{10} \rightarrow n_{15} \\ n_1 \rightarrow n_2 \rightarrow n_5 \rightarrow n_{11} \rightarrow n_{15} \end{cases}$$

$$\mathcal{H}^2 = \begin{cases} n_3 \rightarrow n_5 \rightarrow n_{11} \rightarrow n_{16} \\ n_3 \rightarrow n_6 \rightarrow n_{11} \rightarrow n_{16} \\ n_3 \rightarrow n_6 \rightarrow n_{12} \rightarrow n_{16} \end{cases}$$

$$\mathcal{H}^3 = \begin{cases} n_4 \rightarrow n_3 \rightarrow n_2 \rightarrow n_1 \end{cases}$$

$$\mathcal{H}^4 = \begin{cases} n_{11} \rightarrow n_{10} \rightarrow n_9 \rightarrow n_{13} \\ n_{11} \rightarrow n_{10} \rightarrow n_{14} \rightarrow n_{13} \\ n_{11} \rightarrow n_{15} \rightarrow n_{14} \rightarrow n_{13} \end{cases}$$

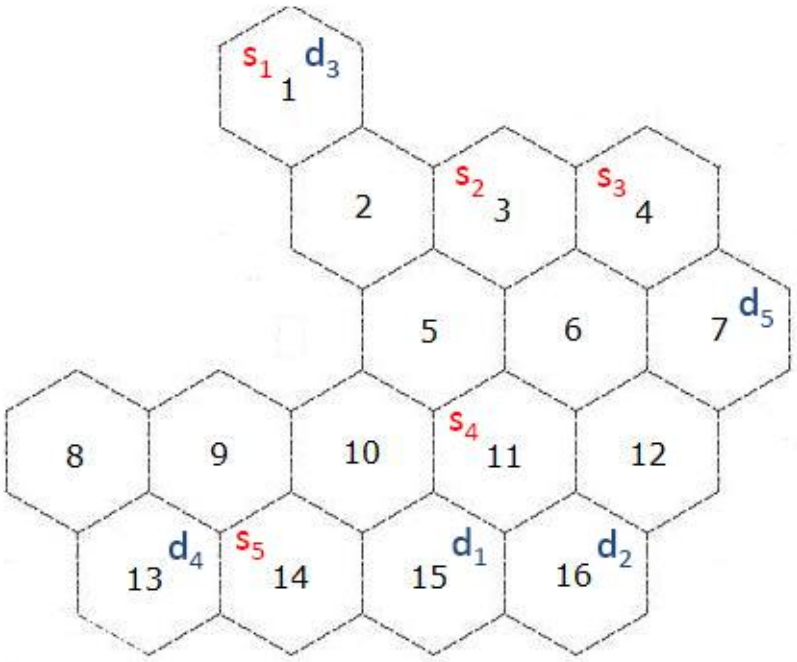


Figure 5–1: The Network Deployment of the Example

$$\mathcal{H}^5 = \begin{cases} n_{14} \rightarrow n_{10} \rightarrow n_5 \rightarrow n_6 \rightarrow n_7 \\ n_{14} \rightarrow n_{10} \rightarrow n_{11} \rightarrow n_6 \rightarrow n_7 \\ n_{14} \rightarrow n_{10} \rightarrow n_{11} \rightarrow n_{12} \rightarrow n_7 \\ n_{14} \rightarrow n_{15} \rightarrow n_{11} \rightarrow n_6 \rightarrow n_7 \\ n_{14} \rightarrow n_{15} \rightarrow n_{11} \rightarrow n_{12} \rightarrow n_7 \\ n_{14} \rightarrow n_{15} \rightarrow n_{16} \rightarrow n_{12} \rightarrow n_7 \end{cases}.$$

In the following subsections, we will show how to model the problem step by step. After each step of modeling, we also show how to deal with the example correspondingly.

5.2.1 Network Topology Matrix – H

Suppose that there are K^s acyclic paths from source s to its destination, represented by an $L \times K^s$ 0–1 matrix $\mathbf{H}^s$ where

$$H^s_{lj} = \begin{cases} 1, & \text{if path } j \text{ of source } s \text{ uses link } l; \\ 0, & \text{otherwise.} \end{cases}$$

Let $\mathcal{H}^s$ be the set of all columns of $\mathbf{H}^s$ that represents all the available paths to s. Define the $L \times K$ matrix $\mathbf{H}$ as

$$\mathbf{H} = [\mathbf{H}^1 \dots \mathbf{H}^S],$$

where $K = \sum_s K^s$ is the total number of paths existing in the network, and $\mathbf{H}$ defines the physical topology of the network.

Given the network topology, it is very easy to generate the topology matrix, and therefore we are not going to show it here.

5.2.2 Flow Fraction Matrix – W

Let $\boldsymbol{w}^s$ be a $K^s \times 1$ vector where the jth entry represents the fraction of flow from s on its jth path such that

$$w^s_j \geq 0 \; \textit{and} \; \mathbf{1}^T \boldsymbol{w}^s = 1,$$

where $\mathbf{1}$ is a vector of an appropriate dimension with the value 1 in every entry. We require $w^s_j \in \{0, 1\}$ for single-path routing. Collecting the vectors $\boldsymbol{w}^s$ for $s = 1, \dots, S$, we get a $K \times S$ block-diagonal matrix $\mathbf{W}$. Let $\mathcal{W}_s$ be the set of all such matrices corresponding to single-path routing defined as

$$\mathcal{W}_s = \{ \mathbf{W} \mid \mathbf{W} = diag(\boldsymbol{w}^1, \dots, \boldsymbol{w}^S) \in \{0, 1\}^{K \times S}, \mathbf{1}^T \boldsymbol{w}^s = 1, \forall s \},$$

where *diag* constructs a matrix given the diagonal elements.[1]

For the given example, if every source s selects the first path in $\mathcal{H}^s$ as its route, then we have $\mathbf{W} = diag(\boldsymbol{w}^1, \ldots, \boldsymbol{w}^5)$ where

$$\begin{aligned} \boldsymbol{w}^1 &= (1,0)^T \\ \boldsymbol{w}^2 &= (1,0,0)^T \\ \boldsymbol{w}^3 &= (1)^T \\ \boldsymbol{w}^4 &= (1,0,0)^T \\ \boldsymbol{w}^5 &= (1,0,0,0,0,0)^T. \end{aligned}$$

5.2.3 Routing Matrix – R

As mentioned above, **H** defines the set of acyclic paths available to each source, and **W** defines how the sources load balance across these paths. Their product defines an $L \times S$ routing matrix $\mathbf{R} = \mathbf{HW}$ that specifies the fraction of the flow of s at each link l. The set of all single-path routing matrices is

$$\mathcal{R}_s = \{\ \mathbf{R} \mid \mathbf{R} = \mathbf{HW},\ \mathbf{W} \in \mathcal{W}_s\ \},$$

[1] When the given elements are vectors, *diag* constructs block diagonal matrix from input argument, which is equivalent to the *blkdiag* command in MatLab. For example, if $\boldsymbol{w}^1 = (1,0)^T$ and $\boldsymbol{w}^2 = (1,0,0)^T$, then

$$diag(\boldsymbol{w}^1, \boldsymbol{w}^2) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

where

$$R_{ls} = \begin{cases} 1, & \text{if link } l \text{ is in the path of source } s; \\ 0, & \text{otherwise.} \end{cases}$$

The routing matrix $\mathbf{R} = \mathbf{HW}$ of the example is very large and is therefore not shown here, but we can write the corresponding routing set as follows:

$$\mathcal{R} = \begin{cases} n_1 \rightarrow n_2 \rightarrow n_5 \rightarrow n_{10} \rightarrow n_{15} \\ n_3 \rightarrow n_5 \rightarrow n_{11} \rightarrow n_{16} \\ n_4 \rightarrow n_3 \rightarrow n_2 \rightarrow n_1 \\ n_{11} \rightarrow n_{10} \rightarrow n_9 \rightarrow n_{13} \\ n_{14} \rightarrow n_{10} \rightarrow n_5 \rightarrow n_6 \rightarrow n_7 \end{cases} .$$

To make it more clear, we draw the corresponding routes on the network, as shown by Figure 5–2.

5.2.4 Network Traffic Matrix – T

We also define an $N \times S$ traffic matrix $\mathbf{T}$ to specify the relationship between routers and sources, where

$$T_{ns} = \begin{cases} 1, & \text{if node } n \text{ is a router for source } s; \\ 0, & \text{otherwise.} \end{cases}$$

In other words, $T_{ns} = 1$ indicates $s \in \mathcal{S}_n$.

A node n is a router for source s if and only if n forwards the data for s. So we define an $N \times L$ matrix $\mathbf{L}_{out}$ to be the out-link matrix which specifies whether a link $l \in \mathcal{L}$ is an

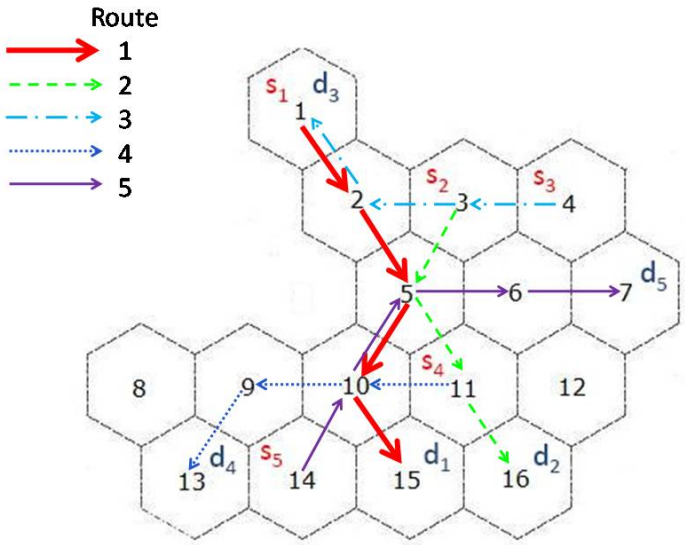


Figure 5–2: The Selected Routes of the Example

out-link of node $n \in \mathcal{N}$, that is

$$L_{nl}^{out} = \begin{cases} 1, & \text{if link } l \text{ is an out-link of node } n; \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, the traffic matrix $\mathbf{T}$ can be calculated as

$$\mathbf{T} = \mathbf{L}^{out}\mathbf{R}.$$

Continuing with the previous example, the corresponding traffic matrix is:

$$\mathbf{T} = \mathbf{L}^{out}\mathbf{R} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

The nth row in $\mathbf{T}$ shows which sources use node n as a router, and it is closely related to the schedulability constraints of node n.

Consider node n_5 as an example. The 5th row of traffic matrix $\mathbf{T}$ is $(1, 1, 0, 0, 1)$, indicating that node n_5 forwards data for sources s_1, s_2 and s_5. According to the schedulability

Table 5–1: Parameters of the Data Sources of the Example

s	α_s	β_s	ω_s	p_s	f_s^{max}	f_s^{min}
1	0.66	0.3	1	0.01	11	30
2	0.66	1.0	2	0.015	2.5	25
3	0.66	0.5	3	0.02	5	30
4	0.66	0.7	4	0.025	1	40
5	0.66	0.3	5	0.03	2	30

Table 5–2: Parameters of the Nodes of the Example

n	1	2	3	4	5	6	7	8
B_n	0.25	0.6	0.4	0.25	0.25	0.7	0.2	0.15
n	9	10	11	12	13	14	15	16
B_n	0.3	0.4	1	0.5	0.25	0.3	0.75	0.3

condition (4.4), we have three inequalities, one for each source:

$$(k_1 f_1 + k_2 f_2 + k_5 f_5) + f_1 \leq \frac{B_5}{l} \tag{5.5}$$

$$(k_1 f_1 + k_2 f_2 + k_5 f_5) + f_2 \leq \frac{B_5}{l} \tag{5.6}$$

$$(k_1 f_1 + k_2 f_2 + k_5 f_5) + f_5 \leq \frac{B_5}{l} \tag{5.7}$$

Therefore, the complete schedulability constraints can be described by the inequality (5.4), with the parameters shown in (5.8).

The parameters of sources are defined in Table 5–1 and the parameters of nodes are defined in Table 5–2. In this paper, the unit for sampling rate is *Hz*, the unit for data block and packet size is *Mb*, and the unit for bandwidth is *Mbps*, if unspecified.

Assume the fixed packet length l is $1kb$, and for simplicity, we assume there is no overhead for the header in this example, i.e., $h = 0$. So, a data block of source s with size

p_s will be divided into $k_s = \lceil \frac{p_s}{l} \rceil$ packets. Table 5–3 shows the number of packets of each source after packet transformation.

$$
\begin{cases}
\mathbf{A} = \begin{pmatrix}
k_1+1 & 0 & 0 & 0 & 0 \\
k_1+1 & 0 & k_3 & 0 & 0 \\
k_1 & 0 & k_3+1 & 0 & 0 \\
0 & k_2+1 & k_3 & 0 & 0 \\
0 & k_2 & k_3+1 & 0 & 0 \\
0 & 0 & k_3+1 & 0 & 0 \\
k_1+1 & k_2 & 0 & 0 & k_5 \\
k_1 & k_2+1 & 0 & 0 & k_5 \\
k_1 & k_2 & 0 & 0 & k_5+1 \\
0 & 0 & 0 & 0 & k_5+1 \\
0 & 0 & 0 & k_4 & 0 \\
k_1+1 & 0 & 0 & k_4 & k_5 \\
k_1 & 0 & 0 & k_4+1 & k_5 \\
k_1 & 0 & 0 & k_4 & k_5+1 \\
0 & k_2+1 & 0 & k_4 & 0 \\
0 & k_2 & 0 & k_4+1 & 0 \\
0 & 0 & 0 & 0 & k_5+1
\end{pmatrix} \\
\boldsymbol{f} = (f_1, f_2, f_3, f_4, f_5)^T \\
\boldsymbol{b} = (\frac{B_1}{l}, \frac{B_2}{l}, \frac{B_2}{l}, \frac{B_3}{l}, \frac{B_3}{l}, \frac{B_4}{l}, \frac{B_5}{l}, \frac{B_5}{l}, \frac{B_5}{l}, \frac{B_6}{l}, \frac{B_9}{l}, \frac{B_{10}}{l}, \frac{B_{10}}{l}, \frac{B_{10}}{l}, \frac{B_{11}}{l}, \frac{B_{11}}{l}, \frac{B_{14}}{l})^T
\end{cases}
\tag{5.8}
$$

Table 5–3: Number of Packets after Packet Transformation

s	1	2	3	4	5
k_n	10	15	20	25	30

Based on the parameters given above, the complete numerical values of constrains (5.2)–(5.4) can be derived and shown in (5.9).

To facilitate distributed computation, we divide the schedulability constraints into individual nodes. The local schedulability constraints at node n are $\mathbf{A}_n \boldsymbol{f} \leq \boldsymbol{b}_n$, where $\mathbf{A}_n$ and $\boldsymbol{b}_n$ define the rows relevant to node n in $\mathbf{A}$ and $\boldsymbol{b}$, respectively. Take node 5 for example again, the schedulability constraints can be shown in the form of $\mathbf{A}_5 \boldsymbol{f} \leq \boldsymbol{b}_5$ with

$$\mathbf{A}_5 = \begin{pmatrix} 11 & 15 & 0 & 0 & 30 \\ 10 & 16 & 0 & 0 & 30 \\ 10 & 15 & 0 & 0 & 31 \end{pmatrix},$$

$$\boldsymbol{b}_5 = (25, 25, 25)^T.$$

Therefore, each node only cares about the local constraints, and the distributed algorithm acts in a way such that all nodes work collaboratively to find the optimal sampling rates which satisfies the local constraints for each node $n \in \mathcal{N}$.

$$
\begin{cases}
\mathbf{A}\boldsymbol{f} = \begin{pmatrix}
11 & 0 & 0 & 0 & 0 \\
11 & 0 & 20 & 0 & 0 \\
10 & 0 & 21 & 0 & 0 \\
0 & 16 & 20 & 0 & 0 \\
0 & 15 & 21 & 0 & 0 \\
0 & 0 & 21 & 0 & 0 \\
11 & 15 & 0 & 0 & 30 \\
10 & 16 & 0 & 0 & 30 \\
10 & 15 & 0 & 0 & 31 \\
0 & 0 & 0 & 0 & 31 \\
0 & 0 & 0 & 26 & 0 \\
11 & 0 & 0 & 25 & 30 \\
10 & 0 & 0 & 26 & 30 \\
10 & 0 & 0 & 25 & 31 \\
0 & 16 & 0 & 25 & 0 \\
0 & 15 & 0 & 26 & 0 \\
0 & 0 & 0 & 0 & 31
\end{pmatrix}
\begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix}
\leq \boldsymbol{b} = \begin{pmatrix}
250 \\ 600 \\ 600 \\ 400 \\ 400 \\ 250 \\ 250 \\ 250 \\ 250 \\ 700 \\ 300 \\ 400 \\ 400 \\ 400 \\ 1000 \\ 1000 \\ 300
\end{pmatrix} \\
\boldsymbol{f} \leq \boldsymbol{f}^{max} = (30, 25, 30, 40, 30)^T \\
\boldsymbol{f} \geq \boldsymbol{f}^{min} = (11, 2.5, 5, 1, 2)^T
\end{cases}
\tag{5.9}
$$

5.3 Constraints

5.3.1 Maximum Device Limit

Although in most cases, the network utility can be increased by raising the sampling rate, unfortunately, the sampling rate is not possible to be raised infinitely. A sensor device may impose a limit on sampling rate due to its physical limitations. We use $\boldsymbol{f}^{max}$ to refer to this limit:

$$\boldsymbol{f} \leq \boldsymbol{f}^{max}.$$

This corresponds to Constraint (5.2).

5.3.2 Minimum Application Requirement

Usually, if a specific application wants to guarantee certain quality of a service, it has to be satisfied with certain level of sampling rate. In other words, an application may impose a minimum sampling rate to maintain its minimum performance level:

$$\boldsymbol{f} \geq \boldsymbol{f}^{min}.$$

This corresponds to Constraint (5.3).

5.3.3 Schedulability Constraint

Based on the analysis in Section 4.3, we can derive the following constraint from the schedulability condition (4.3):

$$\mathbf{A}\boldsymbol{f} \leq \boldsymbol{b}.$$

This corresponds to Constraint (5.4).

CHAPTER 6
Optimizing Sampling Rates with Dynamic Route Selection

In this chapter, we present solutions for the joint optimal sampling rates and route selection problem.

Our solutions draw upon ideas from the recent research of Network Utility Maximization (NUM), which formulates network system design problem as maximization of the aggregate utility of all the nodes subject to physical or economic constraints. Since the publication of the seminal work [21] by Kelly et al. in 1998, the NUM framework has enabled many applications in network research. Compared to the traditional linear network flow problem, the NUM framework takes advantages of many advances in nonlinear optimization theory and distributed algorithms. Specifically, the design of the distributed solution algorithm in this paper is based on the primal-dual method and dual decomposition technique.

6.1 Primal and Dual Problems

The primal problem of the optimal sampling rate assignment with dynamic route selection for real-time wireless sensor network problem is described by (5.1)–(5.4).

A centralized algorithm can be directly derived by solving the primal problem. However, a centralized algorithm requires collecting data from each node. This will generate a lot of traffic and create traffic bottlenecks around the central computing nodes. Due to this reason, a centralized algorithm is inefficient and therefore not suitable for real-life application especially for wireless sensor network application. To avoid this problem, a

distributed algorithm is usually more desirable for solving the optimization problem in sensor networks.

An alternative solution for the optimization problem is based on solving the Lagrangian dual problem corresponding to the primal problem (5.1)–(5.4):

$$\min_{\boldsymbol{\lambda}\geq 0} \sum_{s} \min_{\boldsymbol{f}^{min}\leq \boldsymbol{f}\leq \boldsymbol{f}^{max}} \Big(U_s(f_s) + f_s \min_{\mathbf{R}^s\in\mathcal{H}^s} \sum_{m} A_{ms}\lambda_m\Big) - \sum_{m} b_m\lambda_m, \tag{6.1}$$

where $\boldsymbol{\lambda}$ can be interpreted as the *prices* (or *schedulability prices*) for the schedulability constraints in (5.4). The dual problem (6.1) can be decomposed into several sub-problems, which find the optimal sampling rates and routes in an iterative manner such that the network ULI is minimized and the *schedulability cost* is minimized, as shown later in (6.3) and in (6.4) respectively.

Let $f_s(t)$ be the updated sampling rate proposal for source s at iteration t. Let $\mathbf{R}^s(t)$ be the updated routing vector for source s at iteration t. The dual problem can be solved using dual decomposition, in a manner of gradually approaching:

$$\boldsymbol{\lambda}(t) = \arg\min_{\boldsymbol{\lambda}\geq 0} \sum_{s} \Big(U_s(f_s) + f_s \sum_{m} A_{ms}\lambda_m\Big) - \sum_{m} b_m\lambda_m, \tag{6.2}$$

$$f_s(t) = \arg \min_{\boldsymbol{f}^{min}\leq \boldsymbol{f}\leq \boldsymbol{f}^{max}} U_s(f_s) + f_s \sum_{m} A_{ms}\lambda_m,\ \forall s\in\mathcal{S}, \tag{6.3}$$

$$\mathbf{R}^s(t) = \arg \min_{\mathbf{R}^s\in\mathcal{H}^s} \sum_{m} A_{ms}\lambda_m,\ \forall s\in\mathcal{S}. \tag{6.4}$$

where (6.2) updates the schedulability prices, and (6.3) (6.4) update sampling rates and routing respectively.

An iterative subgradient method can be used to update the prices $\boldsymbol{\lambda}$, which is also referred to as dual variables:

$$\lambda_m(t) = \left[\lambda_m(t-1) + \gamma\Big(\sum_s f_s A_{ms} - b_m\Big)\right]^+, 1 \le m \le M \tag{6.5}$$

where function $[\bullet]^+$ is defined as $[x]^+ = \max\{x, 0\}$, and M is the number of schedulability constraints in constraint set (5.4). Therefore, Equation (6.2) can be replaced by Equation (6.5), and Equations (6.3)–(6.5) form a solution for the dual problem (6.1) by solving the three decomposed sub-problems: price updating, sampling rate assignment and route selection. The sub-problems are coordinated by the prices $\boldsymbol{\lambda}$.

Define the Lagrangian

$$L(\mathbf{R}, \boldsymbol{f}, \boldsymbol{\lambda}) = \sum_s \Big(U_s(f_s) + f_s \sum_m A_{ms}\lambda_m\Big) - \sum_m b_m\lambda_m. \tag{6.6}$$

The primal problem (5.1)–(5.4) and the dual problem (6.1) can be expressed respectively as

$$V_{sp} = \min_{\boldsymbol{f}, \mathbf{R}\in\mathcal{R}_s} \min_{\boldsymbol{\lambda}} L(\mathbf{R}, \boldsymbol{f}, \boldsymbol{\lambda}),$$

$$V_{sd} = \min_{\boldsymbol{\lambda}} \min_{\boldsymbol{f}, \mathbf{R}\in\mathcal{R}_s} L(\mathbf{R}, \boldsymbol{f}, \boldsymbol{\lambda}).$$

Theorem 1

$$V_{sp} \ge V_{sd}.$$

Proof Please refer to Appendix-A. ■

Definition 1 (Duality Gap) *Duality Gap refers to the difference between the optimal value of the primal problem V_{sp} and that of the corresponding dual problem V_{sd}.*

The solution based on Lagrangian dual and dual decomposition works only if there is no duality gap.

6.2 The Centralized Algorithm

A centralized solution can be obtained by solving the sampling rate optimization problem with static routing as shown below for **all** possible route configurations:

$$\min_{\boldsymbol{f}} \sum_{s \in \mathcal{S}} U_s(f_s)$$

subject to

$$\boldsymbol{f} \leq \boldsymbol{f}^{max}$$

$$\boldsymbol{f} \geq \boldsymbol{f}^{min}$$

$$\mathbf{A}\boldsymbol{f} \leq \boldsymbol{b}.$$

Therefore, the optimal routing is the one that minimizes the network ULI, while the solution gives the optimal sampling rates. This optimization problem can be solved with many commercial optimization packages such as the MatLab Optimization Toolbox.[1]

[1] The *fconmin* function from the MatLab Optimization Toolbox can be employed for optimization problems with nonlinear objective function and linear constraints, and one can select the specific solver from the several options coming with the library, including Interior-Point Algorithm, Active Set Algorithm, and Line Search Algorithm. In addition, if an algorithm is not specified manually, MatLab will automatically choose an optimal one based on application characteristics. For our example, the Line Search Algorithm is chosen automatically by MatLab.

6.3 The Distributed Algorithm

In this section, we present our design of the distributed algorithm for solving the optimal sampling rates with dynamic route selection problem. The algorithm is based on the recent research of cross-layer optimization in TCP/IP networks [35], where each constraint in (5.4) is given a schedulability price, and each source tries to select the sampling rate and route that minimize the network ULI and the schedulability cost.

The distributed algorithm has two main attributes:

- It converges to the optimal solution of the optimization problem.
- Each update computation is only based on local information of a node or a source.

First we list some notations that will be used in the distributed algorithm:

γ Step size of updating.

d_n^{out} The worst-case out-degree of a node n, i.e., the data from how many sources are possible to pass through node n, and use n as a router;

b_n^i The right side of the ith schedulability constraint at node n, i.e., $b_n^i = B_n$.

$\boldsymbol{b}_n$ A $d_n^{out} \times 1$ vector with all the elements set to B_n.

λ_n^i The price for the ith constraint at node n.

$\boldsymbol{\lambda}_n$ The vector of schedulability prices at node n.

$\mathbf{A}$ $\mathbf{A} := \mathbf{A}(R, l)$. That is, $\mathbf{A}$ depends on the routing and the pre-defined packet length l, as shown in Section 5.2.

$\mathbf{A}_{ns}\boldsymbol{\lambda}_n$ $\mathbf{A}_{ns}\boldsymbol{\lambda}_n := \sum_{1 \le i \le d_n^{out}} A_{ns}^i \lambda_n^i$.

To facilitate the distributed computation, we divide the M constraints in (5.4) into N sets, each $\boldsymbol{\lambda}_n$ corresponding to a node n. That is, each node only keeps the d_n^{out} schedulability constraints and prices relevant to itself.

The distributed algorithm is made up of an initialization section and an iteration section. In each iteration step, the prices, sampling rates and routes are updated based on the latest information until convergence.

6.3.1 Initialization

a. Each node n sets all the d_n^{out} relevant prices to 1.

b. Each source s sets the sampling rate f_s to f_s^{min}.

c. Each source s selects the first candidate route from $\mathcal{H}^s$.

6.3.2 Update Prices at Iteration t

a. Each source sends out a *RP* (Rate Proposal) packet with the latest rate proposal to its destination along the currently selected route.

b. Upon receiving the *RP* packets from all the relevant sources, each node n computes new prices for the constraints with the following price updating equation:

$$\lambda_n^i(t) = \left[\lambda_n^i(t-1) + \gamma\Big(\sum_s f_s A_{ns}^i - b_n^i\Big)\right]^+, 1 \le i \le d_n^{out}.$$

6.3.3 Update Sampling Rates at Iteration t

a. Each destination sends an *SRU* (Sampling Rate Update) packet with value 0 along the reversed path of the current route to the source.

b. Upon receiving an *SRU* packet, each node adds $\mathbf{A}_{ns}\boldsymbol{\lambda}_n$ to the value in the packet, and forwards it along the reversed path.

c. Upon receiving an *SRU* packet, each source s updates its rate proposal according to local optimization as follows:

$$f_s(t) = \arg \min_{f_s^{min} \le f_s \le f_s^{max}} U_s(f_s) + f_s \sum_n \mathbf{A}_{ns}\boldsymbol{\lambda}_n.$$

6.3.4 Update Routing at Iteration t

a. Each destination sends a *RU* (Routing Update) packet with value 0 along the reversed path of every possible route to the source.

b. Upon receiving a *RU* packet, each node adds $\mathbf{A}_{ns}\boldsymbol{\lambda}_n$ to the value in the packet assuming it is a router for current source s, and forwards the packet along the reversed path.

c. Upon receiving all the *RU* packets from all possible routes for a source-destination pair, each source s updates the routing according to local optimization

$$\mathbf{R}^s(t) = \arg \min_{\mathbf{R}^s \in \mathcal{H}^s} \sum_n \mathbf{A}_{ns}\boldsymbol{\lambda}_n.$$

6.4 Convergence Criteria

Definition 2 (Equilibrium) *We say that ($\tilde{\mathbf{R}}, \tilde{\boldsymbol{f}}, \tilde{\boldsymbol{\lambda}}$) is an equilibrium if it is a fixed point of the above algorithm. That is, starting from routing $\tilde{\mathbf{R}}$, sampling rates $\tilde{\boldsymbol{f}}$ and the associated prices $\tilde{\boldsymbol{\lambda}}$, the algorithm yields ($\tilde{\mathbf{R}}, \tilde{\boldsymbol{f}}, \tilde{\boldsymbol{\lambda}}$) for the next iteration.*

Theorem 2 *An equilibrium ($\tilde{\mathbf{R}}, \tilde{\boldsymbol{f}}, \tilde{\boldsymbol{\lambda}}$) exists if and only if there is no duality gap between the primal problem (5.1)–(5.4) and the dual problem (6.1). In this case, the equilibrium ($\tilde{\mathbf{R}}, \tilde{\boldsymbol{f}}, \tilde{\boldsymbol{\lambda}}$) is a solution for both the primal and dual problems.*

Proof Please refer to Appendix-B. ■

According to *Theorem 2*, the distributed algorithm has an equilibrium exactly when there is no duality gap in the network utility optimization, i.e., when $V_{sp} = V_{sd}$. In other words, when the distributed algorithm converges, the equilibrium found is the solution of the optimization problem (5.1)–(5.4).

In practical applications, the distributed algorithm terminates when an equilibrium is found, which is characterized by the following convergence criteria:

$$\|\boldsymbol{\lambda}(t) - \boldsymbol{\lambda}(t-1)\|_n \leq \varepsilon_\lambda, \tag{6.7}$$

$$\|\boldsymbol{f}(t) - \boldsymbol{f}(t-1)\|_n \leq \varepsilon_f, \tag{6.8}$$

$$\mathbf{R}(t) = \mathbf{R}(t-1), \tag{6.9}$$

where $\varepsilon_\lambda > 0$ and $\varepsilon_f > 0$ are sufficiently small real numbers. $\|v\|_n$ denotes the nth-norm of vector $v = (v_1, \ldots, v_k)$. That is, $\|v\|_1 = \max_i v_i$ and $\|v\|_n = (\sum_{i=1}^{k} v_i^n)^{\frac{1}{n}}$ when $n \in \mathbb{Z}^+$ and $n \neq 1$.

CHAPTER 7
Performance Evaluation

Extensive simulation experiments have been conducted with MatLab and OMNeT++ to demonstrate the efficacy of our solutions. The results and some further analysis are presented in this chapter.

7.1 Convergence

This simulation is based on the parameters of the example in Section 5.2.

First, we solved the optimization problem with the centralized algorithm described in Section 6.2. The optimal network ULI is 0.1877, with $\boldsymbol{f}^* = (22.7273, 10.0000, 11.9048, 11.5385, 9.6775)$ and $\boldsymbol{r}^* = (2, 2, 1, 1, 4)$ where the sth element in $\boldsymbol{r}^*$ indicates the optimal route for source s. For example, the 5th element of $\boldsymbol{r}^*$ is 4, meaning that s_5 should select the 4th route from its candidate route set $\mathcal{H}^5$.

Another simulation experiment was conducted using the distributed algorithm proposed in Section 6.3, with the step size γ set to 0.3. The convergence criteria are described by conditions (6.7)–(6.9), with 2nd-norm, and with ε_λ and ε_f both set to 1×10^{-5}. The algorithm converges within 894 iterations and the result is exactly the same as the one obtained from the centralized solution. The convergence of the distributed algorithm is shown in Figure 7–1 and Figure 7–2, where Figure 7–1 shows the convergence of the global network ULI and Figure 7–2 shows the convergence of the sampling rate of each source.

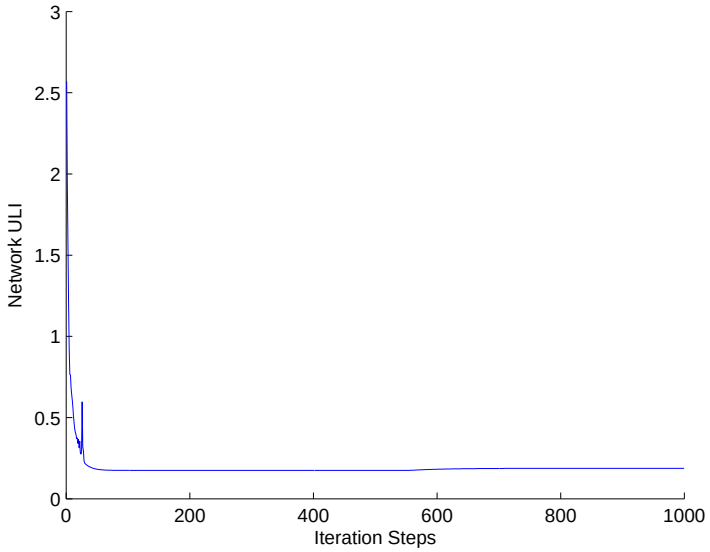


Figure 7–1: The Convergence of Network ULI

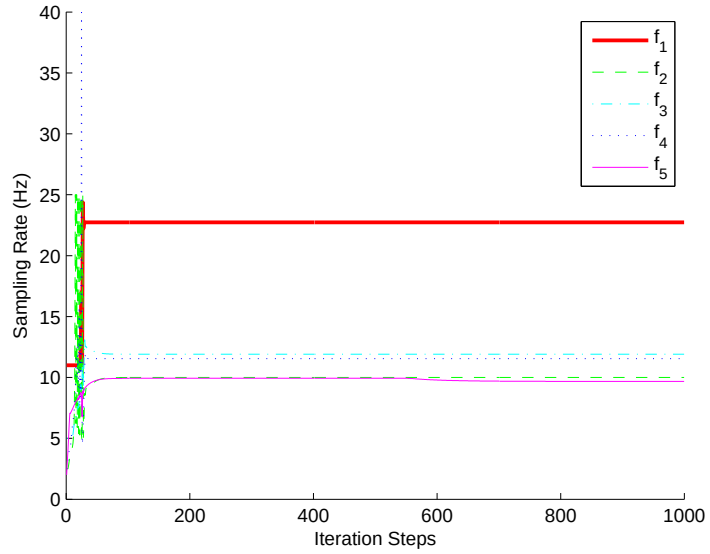


Figure 7–2: The Convergence of Sampling Rates

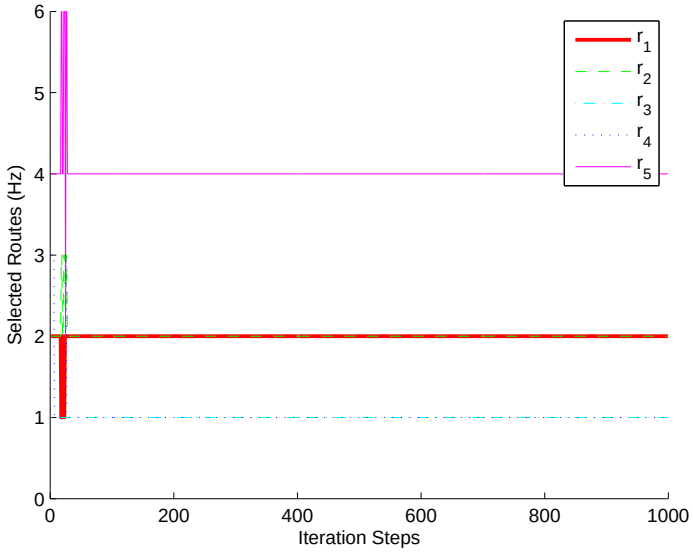


Figure 7–3: The Convergence of Routes

Figure 7–4 shows the leftover bandwidth of each node after the convergence. According to the figure, it is non-negative for every node, indicating that all the nodes are schedulable, since the schedulability condition (4.3) for data transmission model is satisfied.

To demonstrate the advantage of optimizing sampling rates with dynamic route selection, we enumerated all possible routings, solved each corresponding optimization problem with static routing, and then calculated the average network ULI. The average value of network ULI for this experiment is 3.1866, and this shows the importance of dynamic routing: the network utility obtained by optimizing sampling rates with static routing may be very pessimistic for a given static routing, even if the routing is chosen by some existing good routing protocol, because usually a routing protocol is independent with real-time constraints.

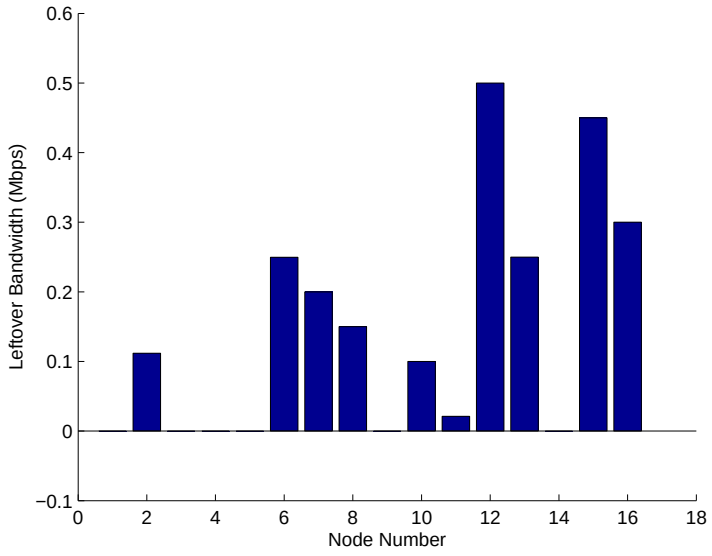


Figure 7–4: The Leftover Bandwidths after Convergence

Table 7–1: Convergence Times with Varied Convergence Thresholds

ϵ	10^{-9}	10^{-8}	10^{-7}	10^{-6}	10^{-5}	10^{-4}
Convergence Iteration	1410	1281	1152	1023	894	766
	10^{-3}	10^{-2}				
	638	501				

Assume $\epsilon_\lambda = \epsilon_f = \epsilon$, we varied the value of the *convergence threshold* ϵ, the number of the iterations needed to achieve the convergence will also be varied. Table 7–1 lists the convergence speed with different value of ϵ. The data in Table 7–5 can be plotted on a figure as shown in Figure 7–5. An interesting observation is that the number of iterations needed for convergence decreases nearly linearly with the increasing of the convergence threshold on the semi-log plot.

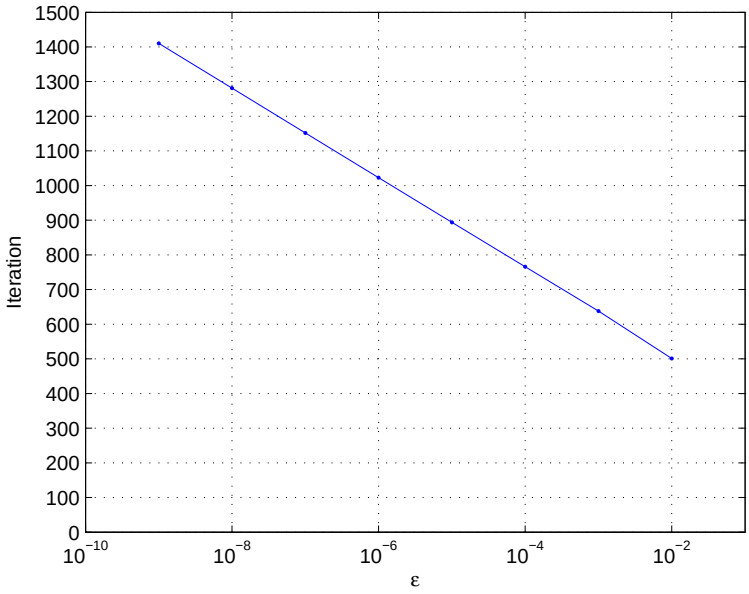


Figure 7–5: Convergence Times with Varied Convergence Thresholds

7.2 Distributed Implementation

Based on the practical parameters of the example, we carried out a simulation with one of the popular network simulators, namely, OMNeT++ [32], [33]. To implement the distributed algorithm, we devised a network protocol that matches the algorithm described in Section 6.3, as follows:

Network Protocol for Distributed Algorithm

The protocol is carried out in iterations, where each iteration consists of 4 steps:

Step 1: Each source sends a *Probe* packet along the currently selected route, and each route updates the constraints upon receiving the *Probe* packet. The *Probe* packet also informs the destination which is the current route.

Step 2: Each source sends out a *RP* (Rate Proposal) packet with the latest rate proposal to its destination along the current route, and each router updates the prices accordingly upon receiving the *RP* packet.

Step 3: Each destination sends a *SRU* (Sampling Rate Update) packet along the reversed path of the current route with a double value set to 0. Upon receiving the packet, each router updates the double value, so that the source is able to update the sampling rate according to Equation (6.3) when receiving the packet.

Step 4: Each destination sends a *RU* (Routing Update) packet along the reversed path of any possible route with a double value set to 0. Upon receiving the packet, each router updates the double value, so that the source is able to decide the route according to Equation (6.4) when receiving all the *RU* packets in this iteration.

A real distributed algorithm has been implemented for OMNeT++ simulation based on the above protocol. We assume each control packet is of 16 bytes, where 4 bytes are used to convey information such as rate proposal in *Step 2* and the double value in *Step 3* and *Step 4*. For the distributed algorithm, we also assume that all the involved routers of the RTWSN are coarse-grain synchronization. This can be achieved, by synchronizing all the nodes and start each step at time kT_{step}, where $k \in \mathbb{Z}$ and T_{step} is the empirical upper bound of end-to-end packet travel time based on the network parameters.

Figure 7–6 is a snapshot of the OMNeT++ simulation under graphic mode. The convergence of sampling rates is shown by Figure 7–7. The obtained sampling rates are consistent with those from MatLab simulation. According to the simulation, the distributed algorithm is able to converge in a short time of no more than 16 seconds.

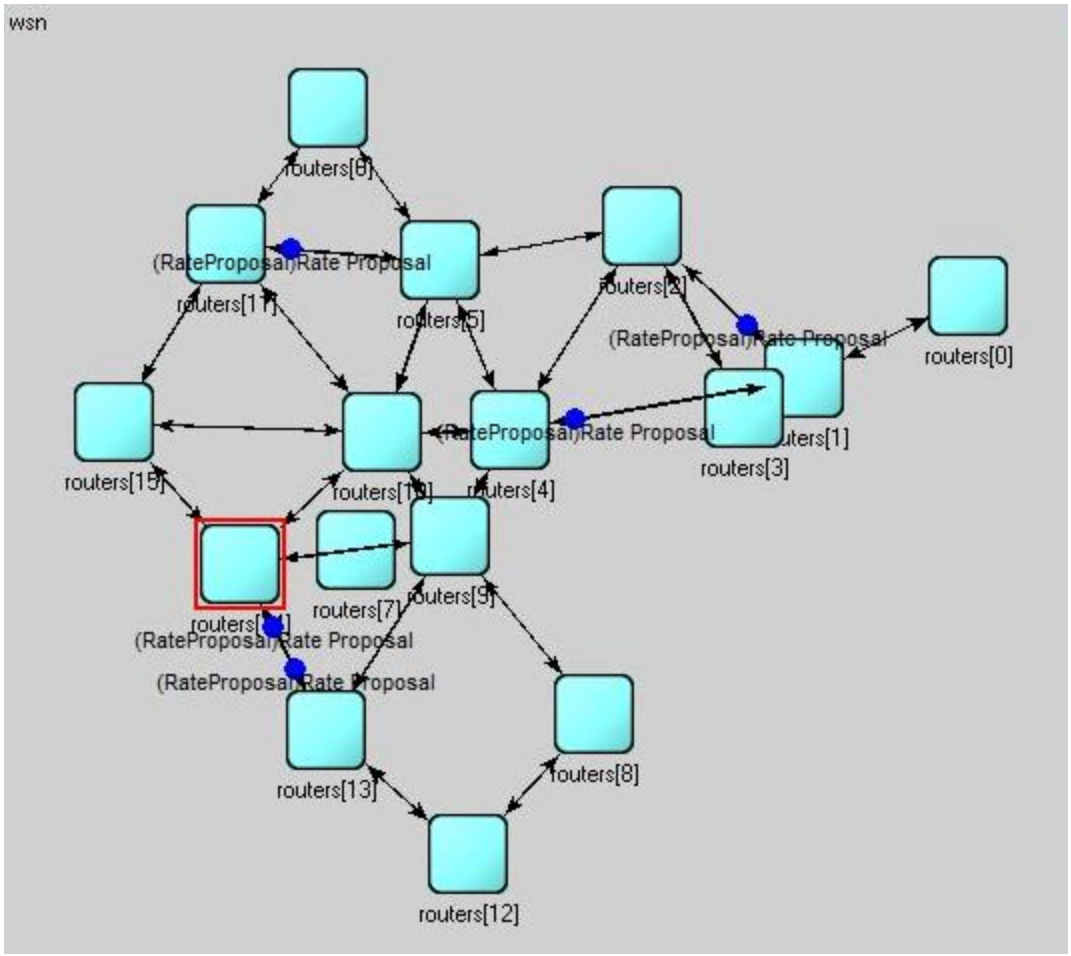


Figure 7–6: A Snapshot of the OMNeT++ Simulation

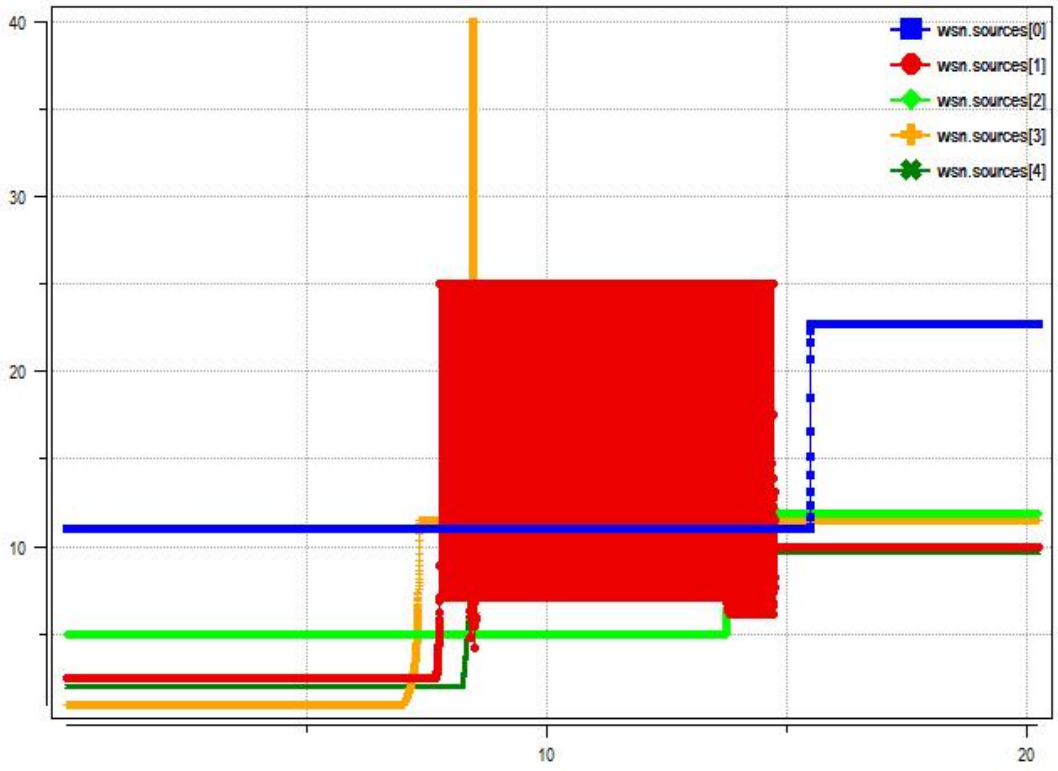


Figure 7–7: The Convergence of Sampling Rates from OMNeT++ Simulation

Table 7–2: Parameters of the Data Sources for the Simulation

s	α_s	β_s	ω_s	p_s	f_s^{max}	f_s^{min}
1	0.66	0.3	1	1	∞	0
2	0.66	1.0	2	1.5	∞	0
3	0.66	0.5	3	2	∞	0
4	0.66	0.7	4	2.5	∞	0
5	0.66	0.3	5	3	∞	0

Table 7–3: Parameters of the Nodes for the Simulation

n	1	2	3	4	5	6	7	8
B_n	16.0	30.0	24.0	16.0	16.0	42.0	12.0	10.0
	9	10	11	12	13	14	15	16
	20.0	24.0	54.0	28.0	16.0	20.0	48.0	14.0

7.3 Effect of Varied Packet Sizes

We also conducted simulation experiments to show how different packet sizes affect the optimal network utility. Assume the header of a packet takes 12 bytes, which is a reasonable value in real applications. Tables 7–2 and 7–3 show the simulation parameters.

The optimal network ULI is 0.8433 if data blocks are not divided into packets, even if the optimization considers the dynamic routing. Based on our packet transformation technique, we varied the packet size that the data blocks are divided into, and the result is shown in Figure 7–8. The achievable optimal network ULI is around 0.5, which is much better than the result of 0.8433 obtained without using the packet transformation technique, i.e., the method in [25].

Another interesting observation from Figure 7–8 is that, the result forms a U-shape: the minimum network ULI is achieved with a medium packet size that is neither too small nor too large. The reason is that, there is a tradeoff between the overhead of transmitting the header information and the utilization waste in schedulability analysis, caused by the

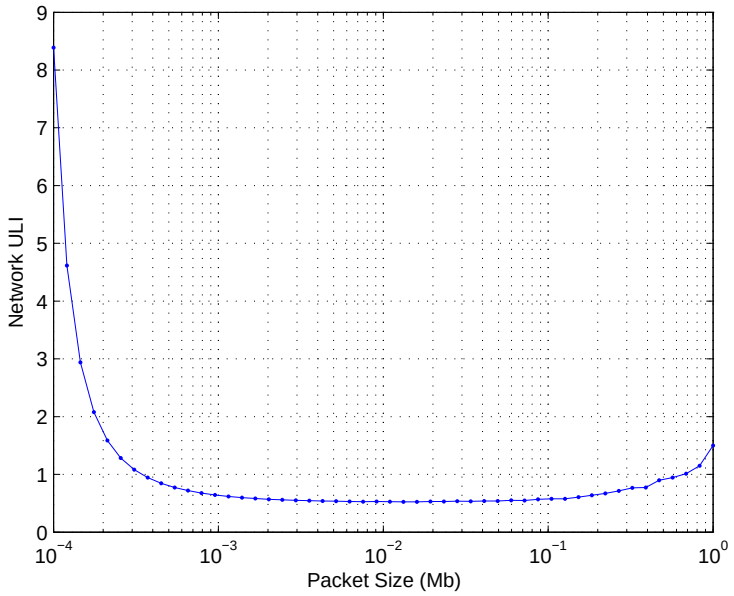


Figure 7–8: The Optimal Network ULI with Different Packet Lengths

worst-case blocking time for sending a packet. A more detailed discussion about the tradeoff can be found in Section 4.3.

7.4 Effect of Step Size γ

The step size of updating γ is an important parameter, which decides how much the prices can be updated and thus affects the convergence speed. If the step size is too small, it may take too much time for the prices to adjust to a certain level, while if it is too large, the prices may not be adjusted slightly to achieve the convergence. Therefore step size must be carefully chosen so that the distributed algorithm is able to converge and converge efficiently.

Figure 7–9 and Figure 7–10 show how the network utility and the sampling rates converge with different values of γ. Figure 7–9(b) shows the case when the step size γ is

properly chosen. Figure 7–9(a) shows the case when the step size is too small, while Figure 7–9(c) shows the case when the step size is too large. Compared to the case shown in Figure 7–9(b) when $\gamma = 0.3$, it takes longer time for the distributed algorithm to converge when $\gamma = 0.1$, while the distributed algorithm cannot converge to one point when $\gamma = 0.5$ due to fluctuation.

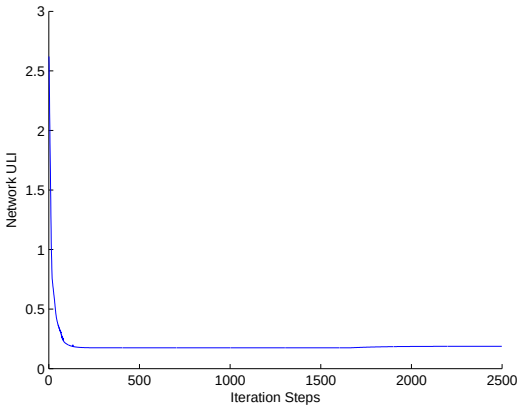
Figure 7–10 shows how individual sampling rates converge with respect to the iteration steps. Figure 7–10(c) indicates that the sampling rates of some sources cannot converge, instead, their values sway within a certain interval. The reason is that, the step size of updating is too coarse for all the sources to decide the optimal routing, therefore, some of the sources will vibrate between two routes with similar schedulability cost. The route selection affects the constraint parameters, which in turn affect the sampling rates selection and cause the fluctuation of the sampling rates.

How to select the initial step size is not the focus of this paper. Interested readers can refer to [25] for more information.

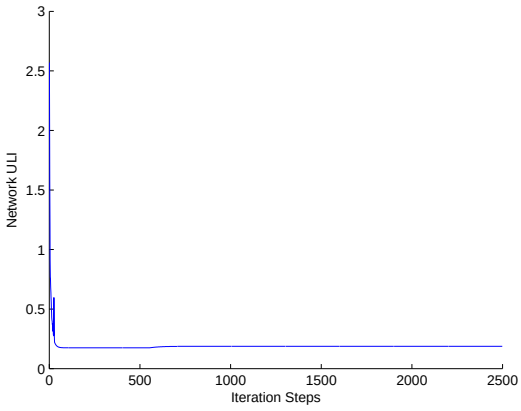
7.5 Incremental Adjustment Property of the Distributed Algorithm

In real world, there are often occasions that ULI functions and constraint set change dynamically. These changes transform the original optimization problem into a new optimization problem with different constraint parameters, hence the optimal sampling rate $\boldsymbol{f}^*$ has to be re-calculated. When distributed algorithm is used, new iterations can be carried out from the existing optimum $(\mathbf{R}^*, \boldsymbol{f}^*, \boldsymbol{\lambda}^*)$, so as to reach the new optimum faster. We call this *incremental adjustment property*. An example is given as follows:

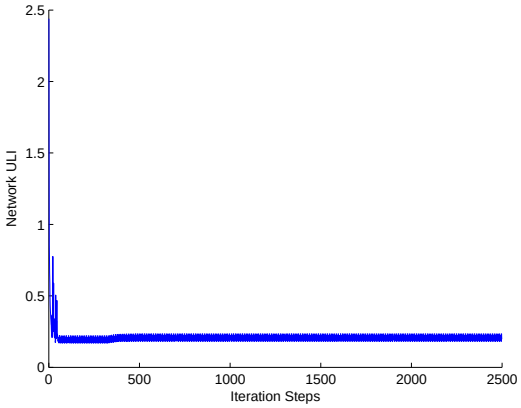
Continue with the simulation example in Section 5.2. Suppose at the 1500th iteration, the ULI coefficients switch from old value set (see Table 5–1) to the new value set



(a) $\gamma = 0.1$

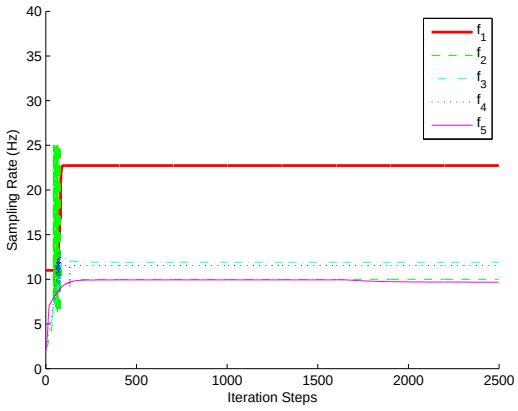


(b) $\gamma = 0.3$

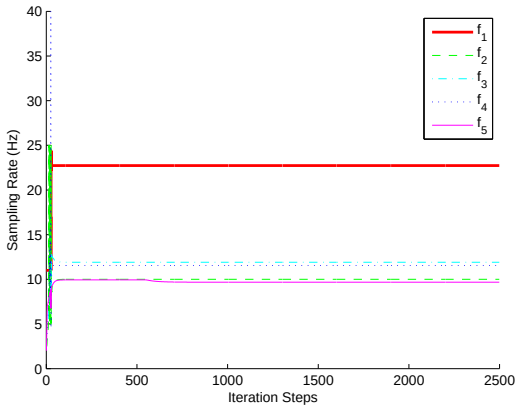


(c) $\gamma = 0.5$

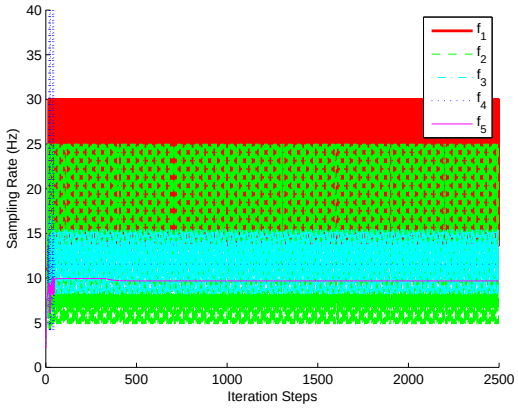
Figure 7–9: The Network ULI Convergence with Different Step Sizes of Updating



(a) $\gamma = 0.1$



(b) $\gamma = 0.3$



(c) $\gamma = 0.5$

Figure 7–10: The Sampling Rates Convergence with Different Step Sizes of Updating

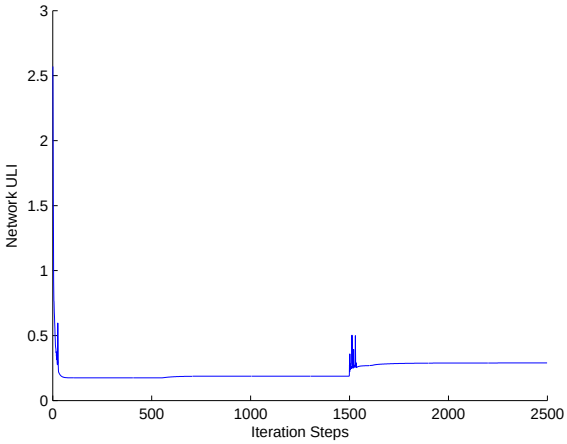
Table 7–4: New Parameters of the Data Sources after the 1500th Iteration

s	α_s	β_s	ω_s	p_s	f_s^{max}	f_s^{min}
1	0.33	0.3	4	0.01	11	30
2	0.22	0.2	3	0.015	2.5	25
3	1.32	0.5	2	0.02	5	30
4	1.98	0.7	1	0.025	1	40
5	0.66	0.3	6	0.03	2	30

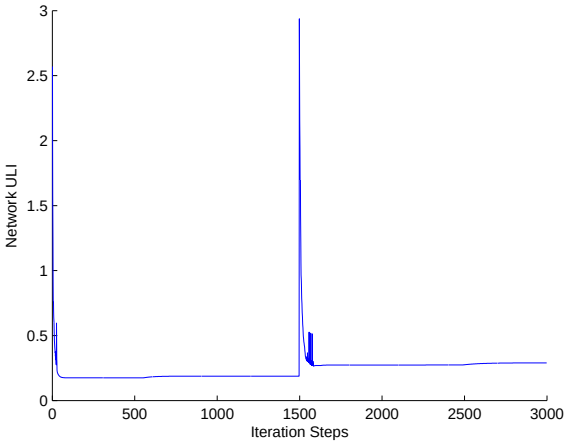
depicted in Table 7–4. Figure 7–11 and Figure 7–12 show the comparison between incremental and non-incremental adjustment schemes: the incremental adjustment scheme starts with the up-to-date optimum $(\mathbf{R}^*, \boldsymbol{f}^*, \boldsymbol{\lambda}^*)$; the non-incremental adjustment scheme starts with the initial configuration of routing, sampling rates and prices, i.e., $(\mathbf{R}^0, \boldsymbol{f}^0, \boldsymbol{\lambda}^0)$, where $\boldsymbol{f}^0 = \boldsymbol{f}^{min}$, $\boldsymbol{\lambda}^0 = \mathbf{1}$, and $\mathbf{R}^0$ is the corresponding routing matrix for the route vector $\boldsymbol{r}^0 = (1, 1, 1, 1, 1)^T$. All other settings of this simulation are the same as those described in Section 5.2. Under incremental adjustment scheme, in about 1000 iterations, the distributed algorithm converges again to the new optimum. In contrast, it takes about another 1500 iterations to converge starting from the initial setting.

7.6 Scalability Analysis for the Distributed and Centralized Algorithms

In this section, the control traffic for both distributed and centralized algorithms are analyzed. The centralized algorithm is efficient when the network is small or intermediate. However, when the network continues to scale up, the centralized algorithm would reach its bottleneck. In contrast, under certain assumptions, the distributed algorithm provides better scalability, though it may be inefficient for very small networks.

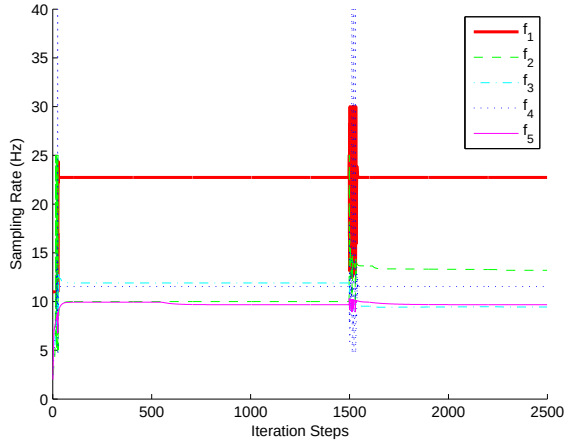


(a) incremental adjustment scheme

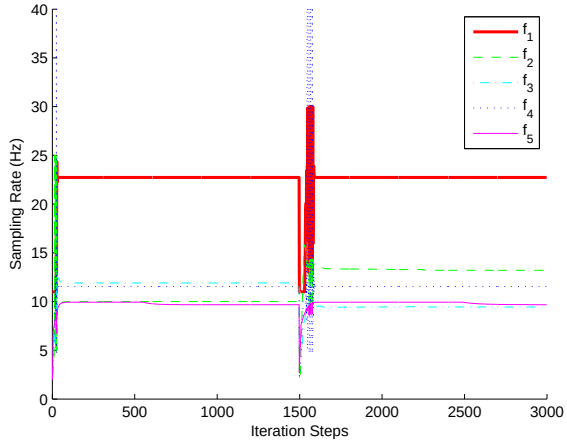


(b) non-incremental adjustment scheme

Figure 7–11: Network Utility Update with Respect to the Number of Iterations



(a) incremental adjustment scheme



(b) non-incremental adjustment scheme

Figure 7–12: Sampling Rates Update with Respect to the Number of Iterations

7.6.1 Control Traffic Analysis for the Distributed Algorithm

Let ϕ_n^{dis} be the accumulated control traffic (in bytes) passing through base station n under the distributed algorithm. Let Φ^{dis} be the maximum accumulated control traffic (in bytes) passing through any of the base stations, i.e., $\Phi^{dis} = \max_{n\in\mathcal{N}}\{\phi_n^{dis}\}$.

Recall that d_n^{out} is the worst-case out-degree of node n, i.e., the number of possible routes passing through node n, and let D_n^{out} be the number of actual routes passing through node n. Thus we have $D_n^{out} \leq d_n^{out} \leq P$, where P the total number of source-destination pairs in the RICH RTWSN.

During each iteration, in *Step 1*, *Step 2* and *Step 3*, D_n^{out} control packets pass through router n respectively. In *Step 4*, a total of d_n^{out} packets pass through router n. Without loss of generality, we assume all control packets are of 16 bytes, in which the header is of 12 bytes, and the payload is of 4 bytes. Therefore the total control traffic passing through each router n during one iteration is $48D_n^{out} + 16d_n^{out}$.

Based on the simulations in the previous sections, the distributed algorithm usually reaches good approximation in 1000 steps. Therefore $\phi_n^{dis} \leq 1000 \times (48D_n^{out} + 16d_n^{out})$. Note that $D_n^{out} \leq d_n^{out} \leq P$, we have $\Phi^{dis} \leq 1000 \times 64 \times P$, i.e., $\Phi^{dis} = O(P)$.

7.6.2 Control Traffic Analysis for the Centralized Algorithm

Let ϕ_n^{cen} be the accumulated control traffic (in bytes) passing through base station n under the distributed algorithm. Let Φ^{cen} be the maximum accumulated control traffic (in bytes) passing through any of the base stations, i.e., $\Phi^{cen} = \max_{n\in\mathcal{N}}\{\phi_n^{cen}\}$.

Let T be the total number possible routing configurations. $T = R_1 \times R_2 \times \ldots \times R_P$, where R_i is the number of candidate routes of source-destination pair i. Under centralized algorithm, for each routing configuration, each source-destination pair needs to send at

least one constraint to the central node. Without loss of generality, we suppose each ULI function is expressed by 3 floating point numbers (12 bytes). To be consistent, we still assume the control packet header is of 12 bytes. Thus the accumulated control traffic payload at the central node c is $\phi_c^{cen} \geq 24PT$, and $\Phi^{cen} = \phi_c^{cen} = \Omega(PT)$.

7.6.3 Comparison between the Distributed and Centralized Algorithms

In small-scale networks, the centralized algorithm is efficient, since the number of source-destination pairs P and the number of possible routing configurations are small. In contrast, the distributed algorithm has to go through hundreds of iterations, or even longer, to converge.

However, in large scale networks, the number of possible routing configurations increase rapidly, and this makes the centralized algorithm impractical. We give an example below to compare the distributed and centralized algorithms.

Assume each source has 3 candidate routes for each destination. According to the discussion in Subsection 7.6.1 and 7.6.2, varying the number of source-destination pairs from 10 to 100, we have the comparison of node traffic between the distributed and centralized algorithms shown in Figure 7–13 as below:

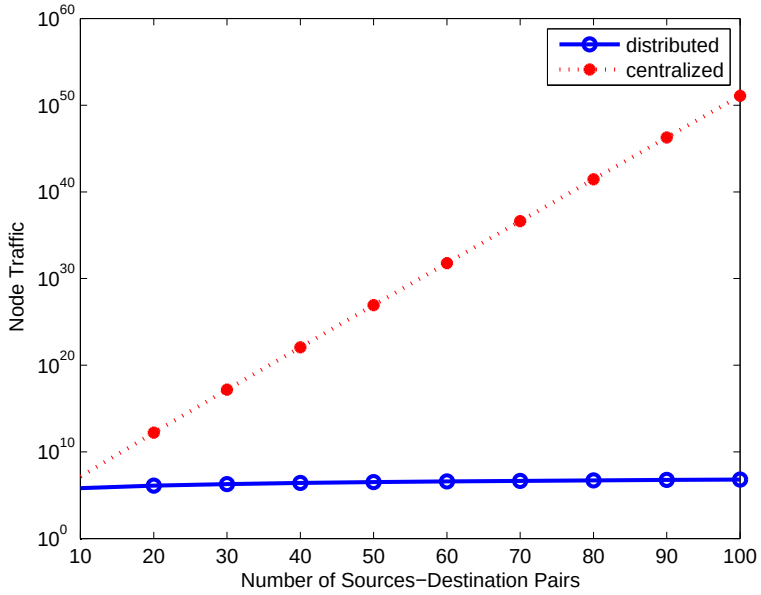


Figure 7–13: Node Traffic with Respect to the Number of Source-Destination Pairs

CHAPTER 8
Conclusion and Future Work

8.1 Conclusion

In this paper, we address the problem of optimizing sampling rate assignment with dynamic route selection for real-time wireless sensor networks. We show that the packet transformation scheme, which involves splitting large data block into smaller packets, can achieve better network utility, especially for applications with large data block sizes. This scheme facilitates schedulability analysis, as well as the implementation of the distributed algorithm for real-time wireless sensor networks. Furthermore, we model the problem as a holistic optimization problem with nonlinear objective function and linear constraints, and present an efficient distributed algorithm based on the primal-dual method. Leveraging the dual decomposition technique, the holistic optimization problem is solved by decomposing the original problem into three sub-problems, which can be solved separately and iteratively. We also demonstrate the efficacy of the distributed algorithm, and the superiority of dynamic routing compared to the static routing counterpart, via extensive simulations.

8.2 Future Work

There are a number of future research directions. First, the convergence speed of the distributed algorithm is dependent upon some design parameters, e.g., the initial prices $\boldsymbol{\lambda}$ and the step size of updating γ, it will be interesting to study how to set these values to maximize the convergence speed. Second, Fig.7–1 and Fig.7–2 show that the utility and

sampling rates take a long time to converge although they are already very close to the equilibrium at a very early stage, and can we make the algorithm converge even faster? Third, we only consider local schedulability for routers in this paper, but it might be desirable to consider end-to-end deadline constraints as well, in order to guarantee end-to-end real-time service. Finally, we plan to implement our algorithm in real-life sensor network testbed with video surveillance or real-time control applications.

Appendix

A. Proof of Theorem 1

In Section 5.2, we have discussed the network model for single-path routing. Multi-path routing does not restrict that the data from a source can be transmitted through only one path. Recall that $\boldsymbol{w}^s$ is a $K^s \times 1$ vector where the jth entry represents the fraction of flow from s on its jth path such that

$$w_j^s \geq 0 \; and \; \mathbf{1}^T \boldsymbol{w}^s = 1.$$

For multi-path routing, we require $w_j^s \in [0, 1]$. The corresponding set $\mathcal{W}_n$ for multi-path routing is

$$\mathcal{W}_m = \{ \mathbf{W} \mid \mathbf{W} = diag(\boldsymbol{w}^1, \ldots, \boldsymbol{w}^S) \in [0,1]^{K \times S}, \mathbf{1}^T \boldsymbol{w}^s = 1, \forall s \}. \tag{1}$$

The set of all multi-path routing matrices is

$$\mathcal{R}_m = \{ \mathbf{R} \mid \mathbf{R} = \mathbf{H}\mathbf{W}, \; \mathbf{W} \in \mathcal{W}_m \},$$

We define:

$$V_{mp} = \min_{\boldsymbol{f}, \mathbf{R} \in \mathcal{R}_m} \min_{\boldsymbol{\lambda}} L(\mathbf{R}, \boldsymbol{f}, \boldsymbol{\lambda}),$$

$$V_{md} = \min_{\boldsymbol{\lambda}} \min_{\boldsymbol{f}, \mathbf{R} \in \mathcal{R}_m} L(\mathbf{R}, \boldsymbol{f}, \boldsymbol{\lambda}).$$

We will prove $V_{sp} \leq V_{sd} = V_{mp} = V_{md}$. Since $\mathcal{R}_s \subseteq \mathcal{R}_m$, $V_{sp} \leq V_{mp}$. We now prove $V_{sd} = V_{md}$ and $V_{mp} = V_{md}$.

Recall that $\mathbf{A}$ depends on variables $\mathbf{R}$ and l, and $\mathbf{R}$ depends on $\mathbf{W}$, we have

$$\begin{aligned} V_{sd} &= \min_{\lambda} \min_{f,\, \mathbf{R}\in\mathcal{R}_s} \sum_s \Big(U_s(f_s) + f_s \sum_m A_{ms}\lambda_m\Big) - \sum_m b_m\lambda_m \\ &= \min_{\lambda} \min_{f} \sum_s \Big(U_s(f_s) + \min_{\mathbf{W}\in\mathcal{W}_s} f_s \sum_m A_{ms}\lambda_m\Big) - \sum_m b_m\lambda_m. \end{aligned}$$

Similarly, for multi-path routing, we have

$$V_{md} = \min_{\lambda} \min_{f} \sum_s \Big(U_s(f_s) + \min_{\mathbf{W}\in\mathcal{W}_m} f_s \sum_m A_{ms}\lambda_m\Big) - \sum_m b_m\lambda_m.$$

Define functions $g_s(\boldsymbol{f}, \boldsymbol{\lambda})$ and $g_m(\boldsymbol{f}, \boldsymbol{\lambda})$ as

$$g_s(\boldsymbol{f}, \boldsymbol{\lambda}) := \min_{\mathbf{W}\in\mathcal{W}_s} f_s \sum_m A_{ms}\lambda_m$$

$$g_m(\boldsymbol{f}, \boldsymbol{\lambda}) := \min_{\mathbf{W}\in\mathcal{W}_m} f_s \sum_m A_{ms}\lambda_m.$$

In order to show that $V_{sd} = V_{md}$, we only need to show that $g_s(\boldsymbol{f}, \boldsymbol{\lambda}) = g_m(\boldsymbol{f}, \boldsymbol{\lambda})$. Clearly $g_s(\boldsymbol{f}, \boldsymbol{\lambda}) \geq g_m(\boldsymbol{f}, \boldsymbol{\lambda})$, since $\mathcal{W}_s \subseteq \mathcal{W}_m$. From (1), noting that $\mathbf{W} = diag(w^i)$, we have

$$g_m(\boldsymbol{f}, \boldsymbol{\lambda}) = \min_{\mathbf{W}} f_s \sum_m A_{ms}\lambda_m$$

subject to

$$\mathbf{1}^T w^i = 1,\ 0 \leq w^i_j \leq 1.$$

Since this is a linear program for the given $\boldsymbol{f}$ and $\boldsymbol{\lambda}$, at least one of the optimal points lies on the boundary, i.e., $w^i_j = 0$ or 1 for all i and j, and hence is in $\mathcal{W}_s \subseteq \mathcal{W}_m$. Such a point solves both $g_s(\boldsymbol{f}, \boldsymbol{\lambda})$ and $g_m(\boldsymbol{f}, \boldsymbol{\lambda})$, i.e., $g_s(\boldsymbol{f}, \boldsymbol{\lambda}) = g_m(\boldsymbol{f}, \boldsymbol{\lambda})$.

V_{mp} is equivalent to the problem

$$\min_{\boldsymbol{f}^{min} \le \boldsymbol{f} \le \boldsymbol{f}^{max},\, \mathbf{R} \in \mathcal{R}_m} \sum_{s \in \mathcal{S}} U_s(f_s) \tag{2}$$

subject to

$$\mathbf{A}\boldsymbol{f} \le \boldsymbol{b}. \tag{3}$$

Note that this is not a convex program since the feasible set specified by $\mathbf{A}\boldsymbol{f} \le \boldsymbol{b}$ is generally not convex.

To show $V_{md} = V_{mp}$, we transform V_{mp} into a convex optimization with linear constraints, which hence has no duality gap. Interested readers can refer to [2] for the relevant basics. Similar to the method shown in [35], by substituting the variables in the problem (2)–(3), we can obtain an equivalent convex program with linear constraint problem as its Lagrangian dual. This indicates $V_{mp} = V_{md}$.

B. Proof of Theorem 2

Necessity

Let $(\tilde{\mathbf{R}}, \tilde{\boldsymbol{f}}, \tilde{\boldsymbol{\lambda}})$ be an equilibrium of the distributed algorithm, then we have

$$\sum_m \tilde{A}_{ms} \tilde{\lambda}_m = \min_{\mathbf{R}^s \in \mathcal{H}^s} \sum_m A_{ms} \tilde{\lambda}_m \,, \ \forall s \in \mathcal{S}, \tag{4}$$

$$(\tilde{\boldsymbol{f}}, \tilde{\lambda}) = \arg\min_{\lambda} \min_{f} \sum_s \Big(U_s(f_s) + f_s \sum_m \tilde{A}_{ms} \lambda_m \Big) - \sum_m b_m \lambda_m. \tag{5}$$

We will show that $(\tilde{\mathbf{R}}, \tilde{\boldsymbol{f}}, \tilde{\boldsymbol{\lambda}})$ solves the dual problem (6.1). Then, since the dual problem lower bounds the primal problem (5.1)–(5.4) (*Theorem 1*), and $\tilde{\mathbf{R}} \in \mathcal{R}_s$ is a single-path routing and hence primal feasible, $(\tilde{\mathbf{R}}, \tilde{\boldsymbol{f}}, \tilde{\boldsymbol{\lambda}})$ also solves the primal problem.

We assume $(\mathbf{R}^*, \boldsymbol{f}^*, \boldsymbol{\lambda}^*)$ is the optimal solution for the dual problem (6.1). That is,

$$\begin{aligned} & (\mathbf{R}^*, \boldsymbol{f}^*, \boldsymbol{\lambda}^*) \qquad (6) \\ = \quad & \arg\min_{\lambda} \min_{f} \sum_s \Big(U_s(f_s) + f_s \min_{\mathbf{R}^s \in \mathcal{H}^s} \sum_m A_{ms} \lambda_m \Big) - \sum_m b_m \lambda_m. \end{aligned}$$

Let

$$g_1(\boldsymbol{\lambda}) := \min_{f} \sum_s \Big(U_s(f_s) + f_s \sum_m \tilde{A}_{ms} \lambda_m \Big) - \sum_m b_m \lambda_m,$$

$$g_2(\boldsymbol{\lambda}) := \min_{f} \sum_s \Big(U_s(f_s) + f_s \min_{\mathbf{R}^s \in \mathcal{H}^s} \sum_m A_{ms} \lambda_m \Big) - \sum_m b_m \lambda_m.$$

Then (5) implies $g_1(\tilde{\boldsymbol{\lambda}}) = \min_{\lambda} g_1(\boldsymbol{\lambda})$, and (6) implies $g_2(\boldsymbol{\lambda}^*) = \min_{\lambda} g_2(\boldsymbol{\lambda})$. Since $\tilde{\mathbf{R}} \in \mathcal{R}_s$, we have

$$g_1(\boldsymbol{\lambda}) \geq g_2(\boldsymbol{\lambda}), \ \forall \boldsymbol{\lambda}$$

and hence

$$g_1(\tilde{\boldsymbol{\lambda}}) = \min_{\lambda} g_1(\boldsymbol{\lambda}) \geq \min_{\lambda} g_2(\boldsymbol{\lambda}) = g_2(\boldsymbol{\lambda}^*).$$

On the other hand

$$\begin{aligned} g_1(\tilde{\boldsymbol{\lambda}}) &= \min_{f} \sum_s \Big(U_s(f_s) + f_s \sum_m \tilde{A}_{ms} \tilde{\lambda}_m \Big) - \sum_m b_m \tilde{\lambda}_m \\ &= \min_{f} \sum_s \Big(U_s(f_s) + f_s \min_{\mathbf{R}^s \in \mathcal{H}^s} \sum_m A_{ms} \tilde{\lambda}_m \Big) - \sum_m b_m \tilde{\lambda}_m \\ &= g_2(\tilde{\boldsymbol{\lambda}}) \\ &\leq g_2(\boldsymbol{\lambda}^*) \end{aligned}$$

where the second equality follows from (4). Therefore $g_1(\tilde{\boldsymbol{\lambda}}) = g_2(\boldsymbol{\lambda}^*) = g_2(\tilde{\boldsymbol{\lambda}})$ and $L(\tilde{\mathbf{R}}, \tilde{\boldsymbol{f}}, \tilde{\boldsymbol{\lambda}}) = L(\mathbf{R}^*, \boldsymbol{f}^*, \boldsymbol{\lambda}^*)$. Moreover, $(\tilde{\mathbf{R}}, \tilde{\boldsymbol{f}}, \tilde{\boldsymbol{\lambda}})$ is an optimal solution of the dual problem.

Sufficiency

Assume that there is no duality gap and $(\mathbf{R}^*, \boldsymbol{f}^*, \boldsymbol{\lambda}^*)$ is an optimal solution for both the primal problem and the dual problem. We claim that it is also an equilibrium of the distributed algorithm. That is, we need to show that

$$\sum_m A^*_{ms}\lambda^*_m = \min_{\mathbf{R}^s \in \mathcal{H}^s} \sum_m A_{ms}\lambda^*_m\,,\ \forall s \in \mathcal{S} \tag{7}$$

and

$$\begin{aligned}(\boldsymbol{f}^*, \boldsymbol{\lambda}^*) &= \arg\min_{\boldsymbol{\lambda}} \min_{\boldsymbol{f}} L(\mathbf{R}^*, \boldsymbol{f}, \boldsymbol{\lambda}) \\ &= \arg\min_{\boldsymbol{f}} \min_{\boldsymbol{\lambda}} L(\mathbf{R}^*, \boldsymbol{f}, \boldsymbol{\lambda}) \qquad (8)\end{aligned}$$

where the equality (8) follows from the assumption that there is no duality gap.

Since $(\mathbf{R}^*, \boldsymbol{f}^*, \boldsymbol{\lambda}^*)$ solves the dual problem (6.1), the optimal routing matrix $\mathbf{R}^*$ satisfies (7) by the Saddle Point Theorem [2]. $(\mathbf{R}^*, \boldsymbol{f}^*, \boldsymbol{\lambda}^*)$ also solves the primal problem (5.1)–(5.4). In particular, $(\boldsymbol{f}^*, \boldsymbol{\lambda}^*)$ solves the utility optimization problem over sampling rates and its Lagrangian dual, with $\mathbf{R}^*$ as the routing matrix, i.e., $(\boldsymbol{f}^*, \boldsymbol{\lambda}^*)$ satisfies (8).

References

[1] M. Andrews. Joint optimization of scheduling and congestion control in communications networks. In *Proceedings of the 40th Conference on Information Sciences and Systems (CISS)*, 2006.

[2] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1 edition, 1995.

[3] J. Bolot, T. Turletti, and I. Wakeman. Scalable feedback control for multicast video distribution in the internet. In *Proceedings of ACM SIGCOMM*, pages 58–67, 1994.

[4] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *Proceedings of the 1st Workshop on Sensor Networks and Applications (WSNA)*, pages 22–31, 2002.

[5] G. C. Butazzo. *Hard Real-Time Computing Systems*. Springer, 2 edition, 2005.

[6] M. Caccamo, L. Y. Zhang, L. Sha, and G. Buttazzo. An implicit prioritized access protocol for wireless sensor networks. In *Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS)*, pages 39–48, 2002.

[7] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle. Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks. In *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM)*, pages 1–13, 2006.

[8] Y. Chen, C. Lu, and X. Koutsoukos. Optimal discrete rate adaptation for distributed real-time systems. In *Proceedings of the 28th IEEE Real-Time Systems Symposium (RTSS)*, pages 181–192, 2007.

[9] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. In *Proceedings of the IEEE*, volume 95, pages 255–312, 2007.

[10] O. Chipara, Z. He, G. Xing, Q. Chen, X. Wang, C. Lu, J. Stankovic, and T. Abdelzaher. Real-time power-aware routing in sensor networks. In *Proceedings of the 14th IEEE International Workshop on Quality of Service (IWQoS)*, pages 83–92, 2006.

[11] R. L. Cruz and A. Santhanam. Optimal routing, link scheduling, and power control in multihop wireless networks. In *Proceedings of the 22th IEEE International Conference on Computer Communications (INFOCOM)*, 2003.

[12] A. Eryilmaz and R. Srikant. Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control. In *Proceedings of the 24th IEEE International Conference on Computer Communications (INFOCOM)*, 2005.

[13] Q. Gao, J. Zhang, X. Shen, and B. Larish. A cross-layer optimization approach for energy efficient wireless sensor networks: coalition-aided data aggregation, cooperative communication, and energy balancing. *Adv. MultiMedia*, 2007(1), 2007.

[14] S. Ghosh, R. Rajkumar, J. Hansen, and J. Lehoczky. Scalable resource allocation for multi-processor qos optimization. In *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS)*, pages 174–183, 2003.

[15] S. Giannecchini, M. Caccamo, and C. Shih. Collaborative resource allocation in wireless sensor networks. In *Proceedings of 16th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 35–44, 30 June-2 July 2004.

[16] J. He, M. Chiang, and J. Rexford. TCP/IP interaction based on congestion price: Stability and optimality. In *IEEE International Conference on Communications (ICC)*, volume 3, pages 1032–1039, 2006.

[17] T. He, J. A. Stankovic, C. Lu, and T. Abdelzaher. SPEED: A stateless protocol for real-time communication in sensor networks. In *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS)*, pages 46–55, 2003.

[18] W. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the 5th International Conference on Mobile Computing and Networking (MobiCom)*, pages 174–185, 1999.

[19] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th International Conference on Mobile Computing and Networking (MobiCom)*, pages 243–254, 2000.

[20] F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[21] F. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: Shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49, 1998.

[22] J. F. Kurose and R. Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Transactions on Computers*, 38(5):705–717, 1989.

[23] C. Lee, J. Lehoczky, D. Siewiorek, R. Rajkumar, and J. Hansen. A scalable solution to the multi-resource qos problem. In *Proceedings of the 20th IEEE Real-Time Systems Symposium (RTSS)*, pages 315–326, 1999.

[24] X. Lin and N. B. Shroff. The impact of imperfect scheduling on cross-layer congestion control in wireless networks. *IEEE/ACM Transactions on Networking*, 14(2):1804–1814, 2006.

[25] X. Liu, Q. Wang, W. He, M. Caccamo, and L. Sha. Optimal real-time sampling rate assignment for wireless sensor networks. *ACM Transactions on Sensor Networks*, 2(2):263–295, 2006.

[26] S. H. Low and D. E. Lapsley. Optimization flow control — I: Basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–874, 1999.

[27] P. Marbach and Y. Lu. Active queue management and scheduling for wireless networks: The single cell case. In *Proceedings of the 40th Conference on Information Sciences and Systems (CISS)*, 2006.

[28] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek. A resource allocation model for qos management. In *Proceedings of the 18th IEEE Real-Time Systems Symposium (RTSS)*, pages 298–307, 1997.

[29] D. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin. On task schedulability in real-time control system. In *Proceedings of the 17th IEEE Real-Time Systems Symposium (RTSS)*, pages 13–21, 1996.

[30] L. Sha, J. Lehoczky, and R. Rajkumar. Solutions for some practical problems in prioritized preemptive scheduling. In *Proceedings of the 7th IEEE Real-Time Systems Symposium (RTSS)*, 1986.

[31] L. Sha, X. Liu, M. Caccamo, and G. Buttazzo. Online control optimization using load driven scheduling. In *Proceedings of the 39th IEEE Conference on Decision and Control (CDC)*, volume 5, pages 4877–4882, 2000.

[32] A. Varga. The omnet++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference*, pages 319–324, June 2001.

[33] A. Varga. OMNeT++. *Column of "Software Tools for Networking", IEEE Network Interactive*, 16(4), July 2002.

[34] C. A. Waldspurger and W. E. Weihl. Lottery scheduling: Flexible proportional-share resource management. In *Proceedings of the 1st USENIX Conference on Operating Systems Design and Implementation (OSDI)*, pages 1–11, 1994.

[35] J. Wang, L. Li, S. H. Low, and J. C. Doyle. Cross-layer optimization in TCP/IP networks. *IEEE/ACM Transactions on Networking*, 13(3):582–595, 2005.

[36] Y. Xi and E. Yeh. Node-based distributed optimal control of wireless networks. In *Proceedings of the 40th Conference on Information Sciences and Systems (CISS)*, 2006.

[37] Y. Xu, J. S. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th International Conference on Mobile Computing and Networking (MobiCom)*, pages 70–84, 2001.

[38] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic. Impact of radio irregularity on wireless sensor networks. In *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 125–138, 2004.