Winter 12-15-2014

Real-Time and Energy-Efficient Routing for Industrial Wireless Sensor-Actuator Networks

Chengjie Wu
Washington University in St. Louis

WASHINGTON UNIVERSITY IN ST. LOUIS

School of Engineering and Applied Science
Department of Computer Science and Engineering

Dissertation Examination Committee:
Chenyang Lu, Chair
Kunal Agrawal
Roger D. Chamberlain
Octav Chipara
Christopher D. Gill
Humberto Gonzalez

Real-Time and Energy-Efficient Routing for Industrial Wireless Sensor-Actuator Networks

by
Chengjie Wu

A dissertation presented to the
Graduate School of Arts and Sciences
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

December 2014
Saint Louis, Missouri

Table of Contents

List of Figures

List of Tables

Acknowledgments

First, I would like to thank my advisor Dr. Chenyang Lu for his continuous guidance and advice both in research and my personal growth. He has introduced me to important research areas, taught me how to find real-world high-impact questions and how to appreciate the beauty and simplicity in real research.

My sincere gratitude goes to my committee members Dr. Kunal Agrawal, Dr. Roger Chamberlain, Dr. Octav Chipara, Dr. Christopher Gill and Dr. Humberto Gonzalez. Thanks for their patient advices on my dissertation.

I am grateful to my current and previous intelligent colleagues in the our research group. My gratitude goes to Greg Hackmann, Yong Fu, Abusayeed Saifullah, Mo Sha, Sisu Xi, Lanshun Nie, Bo Li, Rahav Dor, Jing Li, Dolvara Gunatilaka, Chong Li and Chao Wang.

Finally, I would like to give my deepest gratitude to my parents Keqin Wu and Shufang Wang, my wife Guannan He, and my daughter Ellie Wu. I couldn't be myself without their endless love and support.

Chengjie Wu

Washington University in Saint Louis
December 2014

Dedicated to my parents, my wife, and my daughter

ABSTRACT OF THE DISSERTATION

Real-Time and Energy-Efficient Routing for Industrial Wireless Sensor-Actuator Networks
by
Chengjie Wu
Doctor of Philosophy in Computer Science
Washington University in St. Louis, 2014
Professor Chenyang Lu, Chair

With the emergence of industrial standards such as WirelessHART, process industries are adopting *Wireless Sensor-Actuator Networks (WSANs)* that enable sensors and actuators to communicate through low-power wireless mesh networks. Industrial monitoring and control applications require real-time communication among sensors, controllers and actuators within end-to-end deadlines. Deadline misses may lead to production inefficiency, equipment destruction to irreparable financial and environmental impacts. Moreover, due to the large geographic area and harsh conditions of many industrial plants, it is labor-intensive or dangerous to change batteries of field devices. It is therefore important to achieve long network lifetime with battery-powered devices.

This dissertation tackles these challenges and make a series of contributions. (1) We present a new end-to-end delay analysis for feedback control loops whose transmissions are scheduled based on the Earliest Deadline First policy. (2) We propose a new real-time routing algorithm that increases the real-time capacity of WSANs by exploiting the insights of the delay analysis. (3) We develop an energy-efficient routing algorithm to improve the network lifetime while maintaining path diversity for reliable communication. (4) Finally, we design a distributed game-theoretic algorithm to allocate sensing applications with near-optimal quality of sensing.

Chapter 1

Introduction

With the emergence of industrial standards such as WirelessHART [26] and ISA100 [16], process industries are adopting *Wireless Sensor-Actuator Networks (WSANs)* that enable sensors and actuators to communicate through low-power wireless mesh networks [184]. Industrial process control applications impose stringent end-to-end latency requirements on data communication. To support a feedback control loop, the network periodically delivers data from sensors to a controller and then delivers its control input data to the actuators within an end-to-end deadline. Consequences of deadline misses in data communication may range from production inefficiency, equipment destruction to irreparable financial and environmental impacts.

To meet the stringent real-time performance requirements of control systems, there is a critical need for fast end-to-end delay analysis for real-time flows that can be used for online admission control. We present a new end-to-end delay analysis for periodic flows whose transmissions are scheduled based on the Earliest Deadline First (EDF) policy. Our analysis comprises novel techniques to bound the communication delays caused by channel contention and transmission conflicts in a WSAN. Furthermore, we propose a technique to reduce the pessimism in admission control by iteratively tightening the delay bounds for flows with short deadlines. Experiments on a WSAN testbed and simulations demonstrate the effectiveness of our analysis for online admission control of real-time flows.

Routing has significant impacts on reliability, real-time capacity and network lifetime. The core contributions of this dissertation tackles the real-time communication and network lifetime problems in WSAN routing. We first design real-time routing algorithms that leverage the insights from the delay analysis. By incorporating conflict delays in the routing decisions,

our real-time routing algorithms allow WSANs to accommodate more feedback control loops while meeting their deadline constraints.

Our second contribution to routing addresses the energy constraints of field devices in WSANs. Since many industrial devices operate on batteries in harsh environments where changing batteries are labor-intensive, WSANs need to achieve long network lifetime. To meet industrial demand for long-term reliable communication, we propose efficient graph routing designs to maximize network lifetime of WSANs. We first formally formulate the network lifetime maximization problem for WSANs under graph routing and prove it is NP-complete. We then propose the optimal algorithm and two more efficient algorithms to prolong the network lifetime of WSANs. Experiments in a physical testbed and simulations show our linear programming relaxation and greedy heuristics can improve the network lifetime by up to 50% while preserving the reliability benefits of graph routing.

Besides industrial WSANs, we have seen wireless sensor networks built as an integrated infrastructure shared by multiple environmental monitoring applications. Given the resource constraints of sensor devices, it is important to optimize the allocation of applications to maximize the overall quality of sensing. Recent solutions to this challenging application allocation problem are centralized in nature, limiting their scalability and robustness against network failures and dynamics. We present a distributed game-theoretic approach to allocate monitoring applications. We first transform the application allocation problem to a submodular game and then develop a decentralized algorithm that only employs localized interactions among neighboring devices. We prove that the network can converge to a pure strategy Nash equilibrium with an approximation bound of 1/2. Simulations based on three real-world datasets demonstrate that our algorithm is competitive against a state-of-the-art centralized algorithm in terms of quality of sensing.

The rest of this dissertation is organized as follows. Chapter 2 introduces the WSAN system model we used in our dissertation. Chapter 3 presents our end-to-end delay analysis for Earliest Deadline First (EDF) scheduling policy. Chapter 4 presents our real-time routing design. Chapter 5 presents our energy-efficient graph routing design. Chapter 6 discuss our distributed application allocation design in shared sensor networks. Chapter 7 concludes this dissertation.

Chapter 2

Network Model of Wireless Sensor-Actuator Networks

We consider a WSAN architecture based on the WirelessHART standard [26]. A WSAN (as shown in Figure 2.1) consists of a gateway, multiple access points, and a set of field devices. The gateway is wired to the access points. The access points and network devices are all equipped with half-duplex radio transceivers compatible with the IEEE 802.15.4 physical layer. The gateway communicates with field devices, such as sensors or actuators, through the access points. The access points and the field devices form a wireless mesh network. We use network device to refer any device in the system, including the gateway, an access point and a field device.

The WSAN adopts centralized network management, where a network manager (usually running in the gateway) manages all devices. The network manager gathers the network topology information from the network devices, and generates and disseminates the routes and transmission schedule to all network devices. This centralized network management architecture, adopted by the WirelessHART standard, enhances the predictability and visibility of network operations at the cost of scalability.

The WSAN adopts a Time Division Multiple Access (TDMA) MAC layer protocol on top of the IEEE 802.15.4 physical layer. All devices across the network are synchronized. Time is divided into 10 ms slots, and each time slot can accommodate one data packet transmission and its acknowledgment. The WSAN supports multi-channel communication using channels defined in the IEEE 802.15.4 standard. Only one transmission is scheduled on each channel across the whole network to avoid potential collision between concurrent transmissions in a

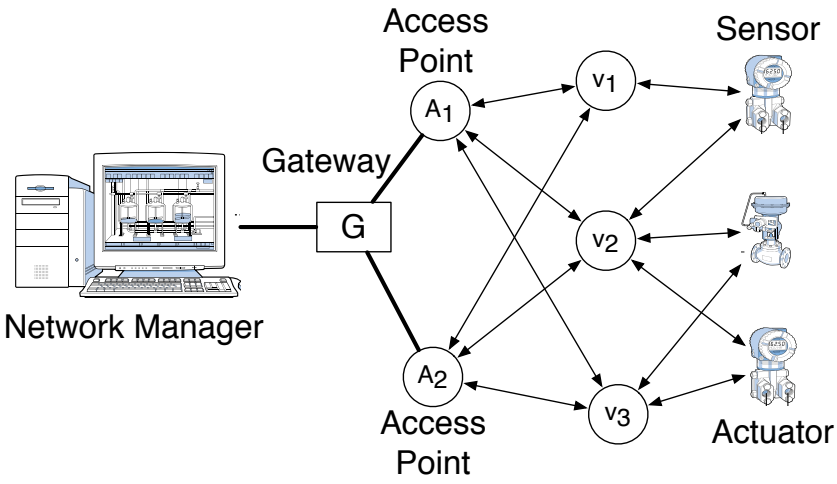


Figure 2.1: Topology of a typical WSAN

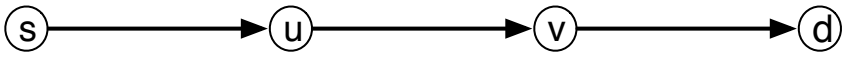
same channel. While this conservative design reduces network throughput and scalability, it avoids interference between transmissions within the network and thereby enhances reliability and predictability, which are important for industrial applications.

2.1 Routing Model

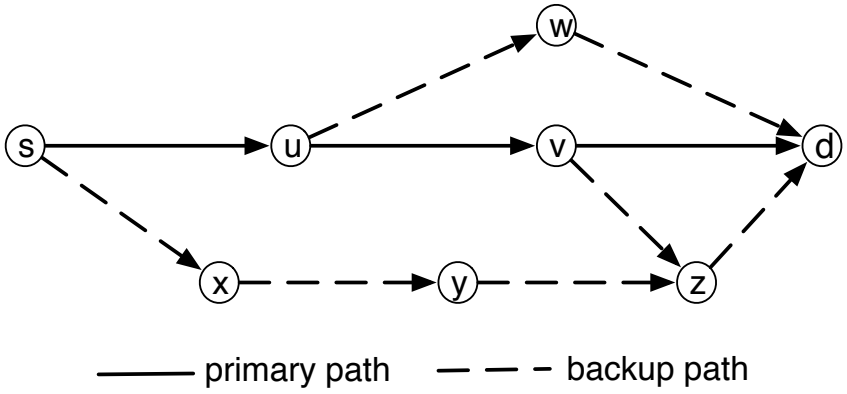
As specified in the WirelessHART standard, WSANs adopt both source routing and graph routing. In source routing, a single path from the source to the destination is used to deliver packets, as shown in Figure 2.2(a).

In graph routing, redundant routes are provided to handle link failures, as shown in Figure 2.2(b). In a graph route, a single path is used as primary path (solid arrows in Figure 2.2(b)). For each network device on the primary path, except the destination, a backup path (dashed arrows in Figure 2.2(b)) is provided to handle link failures. For example, backup path $u \rightarrow w \rightarrow d$ is built to handle the failure of link $\overrightarrow{uv}$.

We study a network topology $G = (V, E)$ as a set of network devices V and directed links between network devices E. A link here can be a wireless link between two field devices, a



(a) Source Route



(b) Graph Route

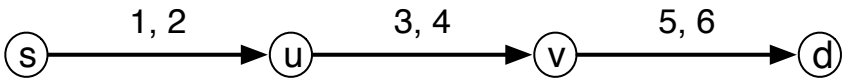
Figure 2.2: Source and Graph Routing

wireless link between an access point and a field device, or a wired link between an access point and the gateway. We define a graph route as follows:

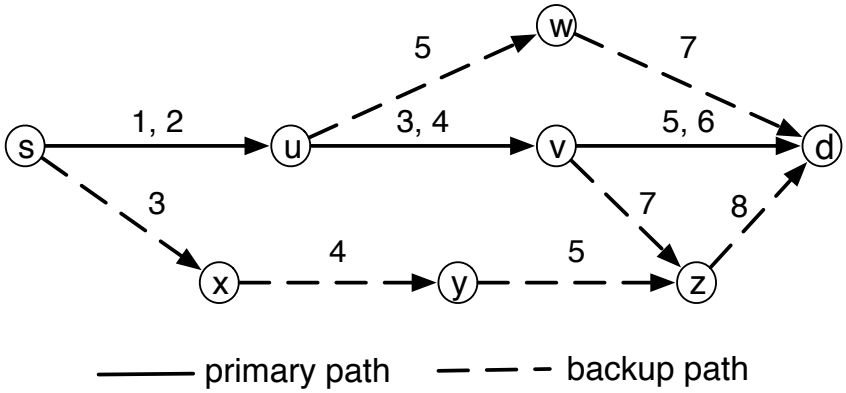
Definition 1. *Given a source s and a destination d, a graph route $R = \{\phi_0, \phi_1, \cdots, \phi_k\}$ is a set of paths from s to d. ϕ_0 is the primary path. Every network device v_i on the primary path ϕ_0, except the destination d, has a backup path ϕ_i from itself to the destination which does not include v_i's outgoing link on the primary path.*

Clearly, a graph route can tolerate any single link failure. If a link on the primary path fails, there is a backup path to tolerate this failure.

A WSAN can support multiple process control loops, each of which introduces a periodic data flow in the network. Each flow has a period, a sensor, and an actuator. For each flow, there are two graph routes: an uplink graph route and a downlink graph route. The uplink graph route starts from the sensor and ends at the gateway. A downlink route starts from the gateway and ends at the actuator.



(a) Transmission scheduling on a source route



(b) Transmission scheduling on a graph route

Figure 2.3: Transmission Scheduling Examples

2.2 Transmission Scheduling Model

In a WSAN, a time slot can be a dedicated slot or a shared slot. In a dedicated slot, only one transmission is scheduled. However, in a shared slot, multiple contending transmissions can be scheduled.

In source routing, only dedicated slots are used. As shown in Figure 2.3(a), one transmission and one retransmission are scheduled on dedicated slots for each link on the source route to handle a transmission failure.

In graph routing, both dedicated slots and shared slots are used. For each device on the primary path, the network manager allocates two dedicated slots for a transmission and a retransmission on its outgoing link on the primary path, followed by a third shared slot on its outgoing link on its backup path. Thus, each link on the primary path is assigned two dedicated slots and links on backup paths are assed to shard slots. Since a WSAN usually employs only high-quality links, shared slots are therefore assigned to backup paths to reduce delay and enhance bandwidth

We show an example of scheduling on a graph route in Figure 2.3(b). Each transmission on the primary path is scheduled twice on dedicated slots, and each transmission on the backup path is scheduled only once on a shared slot. Note we use three channels in this example, so transmissions $\overrightarrow{vd}, \overrightarrow{uv}$, and $\overrightarrow{yz}$ can be scheduled in the same time slot 5 on three different channels.

Chapter 3

Delay Analysis of EDF Scheduling for Wireless Sensor-Actuator Networks

Industry is adopting Wireless Sensor-Actuator Networks (WSANs) as the communication infrastructure for process control applications. To meet the stringent real-time performance requirements of control systems, there is a critical need for fast end-to-end delay analysis for real-time flows that can be used for online admission control. This chapter presents a new end-to-end delay analysis for periodic flows whose transmissions are scheduled based on the Earliest Deadline First (EDF) policy. Our analysis comprises novel techniques to bound the communication delays caused by channel contention and transmission conflicts in a WSAN. Furthermore, we propose a technique to reduce the pessimism in admission control by iteratively tightening the delay bounds for flows with short deadlines. Experiments on a WSAN testbed and simulations demonstrate the effectiveness of our analysis for online admission control of real-time flows.

3.1 Introduction

With the emergence of industrial standards such as WirelessHART [26] and ISA100 [16], process control industries are adopting *Wireless Sensor-Actuator Networks (WSANs)* in which sensors and actuators communicate through low-power multi-hop wireless mesh networks [184]. Since excessive communication delay may lead to severe degradation of control performance or even instability of the control system, it is critical to estimate worst-case end-to-end communication delays for real-time flows in WSANs [182]. Moreover, fast delay

analysis is needed for online admission control and network reconfiguration in response to dynamic changes of channel conditions in industrial environments.

In this chapter, we present a new delay analysis for periodic flows in WSANs in which transmissions are scheduled based on the *Earliest Deadline First (EDF)* policy, a common real-time scheduling policy that has been found to be an effective transmission scheduling policy for real-time WSANs in recent studies [172].

Our new delay analysis can be used to derive end-to-end delay bounds for real-time flows in WSANs. The key feature of our analysis lies in a novel approach to combine two types of delays in a WSAN: contention delays due to limited number of wireless channels, and conflict delays caused by conflicts among concurrent wireless transmissions involving a same device. Furthermore, we reduce the pessimism in admission control by iteratively tightening the delay bounds for flows with short deadlines.

We evaluate our delay analysis through experiments on a 63-node WSAN testbed and simulations. The experiment results demonstrate our delay analysis provides safe bounds of real end-to-end delays. The simulation results show that our delay analysis is effective in term of acceptance ratio when used for admission control. We also provide a comprehensive simulation study that compares a state-of-the-art fixed priority scheduling algorithm [171] and a dynamic priority scheduling algorithm [172]. Our simulations show EDF outperforms fixed priority scheduling [171] in term of real-time performance, while delivering competitive acceptance ratios to the existing dynamic priority scheduling policy at lower computational cost.

The rest of the chapter is organized as follows. Section 3.2 reviews related works. Section 3.3 describes the EDF scheduling policy. Section 3.4 presents the delay analysis. Section 3.5 evaluates our delay analysis through experiments and simulations. Section 3.6 concludes the chapter.

3.2 Related Works

Real-time transmission scheduling in wireless sensor networks received considerable attention [186]. In contrast to previous works on traditional sensor networks, our research investigates real-time WSANs based on recent industrial standards such as WirelessHART with unique features. For example, our analysis is designed for wireless mesh networks running a multi-hop and multi-channel TDMA protocol. In contrast, the network model in previous works is based on single channel [27,59,61,107,151,155], or CSMA/CA MAC protocol [151,173]. While some earlier works [27,59,61,107,155] analyze fixed priority scheduling in wireless networks, we focus on EDF scheduling, which is a dynamic priority scheduling policy. The probabilistic delay analyses for EDF proposed in [104] are not suitable for industrial WSANs that require safe bounds on network delays. Earlier efforts on real-time schedulability analysis for EDF [45,46] adopt a cellular network structure and require wireless nodes with full-duplex transceivers.

In the area of industrial WSANs, earlier works study the transmission scheduling for WSANs with simple topologies such as linear [214], tree [180, 211] and cluster tree topologies [192]. Transmission scheduling of real-time flows for arbitrary WSAN topologies has been studied in [172]. It presents a real-time scheduling algorithm based on branch-and-bound and a dynamic priority scheduling algorithm called C-LLF. However, it does not present any delay analysis to derive its delay bound, therefore requires laying out the entire transmission schedule of the whole network, which incurs high computation delays in admission control. Near optimal rate selection for fixed priority scheduling has been studied in [167,168]. End-to-end delay analysis for fixed priority scheduling in WSANs has been proposed in [169,170]. The performance of fixed priority scheduling highly depends on the priority assignment, which is proven to be a difficult problem, and near-optimal priority assignment algorithms incur significant computational cost when used online.

While dynamic priority scheduling represents an attractive alternative to fixed priority scheduling, end-to-end delay analysis for dynamic priority scheduling has not been studied for WSANs. Our work provides an end-to-end delay analysis for EDF scheduling policy, which is a commonly used dynamic priority scheduling algorithm in real-time systems [32, 38, 39] and outperforms the state-of-the-art fixed priority scheduling algorithms in WSANs in our simulation study.

3.3 EDF Scheduling

The WirelessHART standard supports two types of routing: source routing and graph routing. Source routing provides a single route for each flow, whereas graph routing provides multiple redundant routes in a routing graph and therefore enhances reliability through route diversity. Our analysis currently assumes source routing and can be easily extended to a model where each flow has multiple source routes and send redundant packets through every route to enhance reliability. Supporting graph routing is part of our future work.

We consider a set of periodic flows $F = \{F_1, F_2, \cdots, F_N\}$ to be scheduled on m channels. Each flow

$$F_k = (D_k, T_k, \alpha_k, \phi_k, C_k)$$

is characterized by a relative deadline D_k, a period T_k, a start time α_k, a route ϕ_k and an transmission count C_k. The route ϕ_k is composed of a sequence of links in the network from the source device s_k to the destination device a_k. To enhance reliability, at most κ (re)transmissions are scheduled for one link. Once the sender receives the acknowledgment, it will discard other retransmission retries. The transmission count C_k equals the total number of (re)transmissions scheduled for one packet of this flow along its route, i.e., $C_k = |\phi_k|\kappa$, where $|\phi_k|$ is the length of ϕ_k.

We follow the constrained deadline model where the deadline of each flow is within its period, i.e., $D_k \leq T_k$. Hence different packets of the same flow cannot co-exist in the network in the same time slot. For flow F_k, a new packet is released at source node s_k in the beginning of each period. We use $P_{k,j}$ to refer to the j^{th} packet of the flow F_k, whose release time is $r_{k,j} = \alpha_k + (j-1)\,T_k$. Packet $P_{k,j}$ needs to be delivered to the destination a_k through a sequence of transmissions along ϕ_k. If $P_{k,j}$ is delivered to the destination at slot $f_{k,j}$ through its route, its *end-to-end delay* $R_{k,j}$ is $f_{k,j} - r_{k,j} + 1$. A packet needs to complete all its transmissions before its absolute deadline $d_{k,j} = r_{k,j} + D_k$. We use R_k to denote the end-to-end delay of flow F_k, which is the maximum end-to-end delay of all its packets.

The network manager generates schedules for all field devices up to the hyper-period, i.e., the least common multiply of $\{T_k, k = 1, \ldots, n\}$. When generating schedules, the network manager follows the EDF scheduling policy. For all released packets, each packet is assigned a priority based on its absolute deadline. The packet with an earlier absolute deadline is

assigned a higher priority. At any time slot, if there remains an available channel, among all released but not delivered packets which do not conflict with packets already scheduled in this time slot, the packet with highest priority is scheduled to this slot. This process repeats until all channels are occupied or all remaining packets conflict with at least one scheduled packet. Transmissions of the same packet can be scheduled on different channels at different time slots.

3.4 Worst-Case End-to-End Delay Analysis

In this section, we present our worst-case end-to-end delay analysis for real-time flows under the EDF policy. A set of real-time flows is schedulable if every flow has a worst-case end-to-end delay that is no greater than its deadline. Given a set of real-time flows, our goal is to derive an upper bound on the worst-case end-to-end delay of every flow. The delay analysis can be used as a schedulability test of the flow set under EDF.

3.4.1 Terminology

Before analyzing the delays, we first introduce the terminology used in the analysis. We say a packet is *ready* if it is released and not delivered yet. We say a packet *executes* in a time slot if it has a transmission scheduled in this time slot. A packet can be delayed for two reasons.

- **Conflict delay:** Due to the half-duplex radio, two transmissions conflict with each other if they share a node (sender or receiver). Then only one of them can be scheduled at current time slot. Therefore, if a packet conflicts with another packet that has already been scheduled in the current time slot, it has to be postponed to a later time slot, resulting in conflict delay.
- **Contention delay:** As a WSAN does not allow concurrent transmissions in a same channel, each channel can only accommodate one transmission across the network in each time slot. If all channels are assigned to transmissions of other packets, a packet must be delayed to a later slot, resulting in contention delay.

To be more precise, we define the *conflict delay* of packet $P_{k,j}$ as the number of time slots when packet $P_{k,j}$ is delayed because it conflicts with higher priority packets. We denote conflict delay of packet $P_{k,j}$ as $Y^f_{k,j}$. We define the *contention delay* of packet $P_{k,j}$ as the number of time slots when $P_{k,j}$ is delayed because all the channels are occupied by higher priority packets and none of them conflict with $P_{k,j}$. We denote contention delay of packet $P_{k,j}$ as $Y^t_{k,j}$. Then the end-to-end delay of packet $P_{k,j}$ is

$$R_{k,j} = Y^f_{k,j} + Y^t_{k,j} + C_k, \tag{3.1}$$

where C_k is the transmission count of $P_{k,j}$ along its route.

We define the *interference* of a flow F_l on packet $P_{k,j}$ as the number of slots when $P_{k,j}$ waits for transmissions of packets belonging to F_l. We denote flow F_l's interference on packet $P_{k,j}$ as $I_{k,j}(l)$. Note the terminology *interference* refers to the time a packet is delayed by transmissions associated with another flow. It is not related to interference between concurrent wireless transmissions, which cannot occur in a WSAN because it does not allow concurrent transmissions in a same channel. We further categorize flow F_l's interference on packet $P_{k,j}$ into two: *conflict interference* $I^f_{k,j}(l)$ and *contention interference* $I^t_{k,j}(l)$. Flow F_l's conflict interference on packet $P_{k,j}$ is the number of time slots when $P_{k,j}$ is delayed due to conflicting transmissions belonging to flow F_l. F_l's contention interference on $P_{k,j}$ is the number of time slots when $P_{k,j}$ waits while transmissions of flow F_l are executed and do not conflict with $P_{k,j}$. By definition, we have

$$I_{k,j}(l) = I^f_{k,j}(l) + I^t_{k,j}(l). \tag{3.2}$$

In the rest of this section, we present the worst-case delay analysis in following 4 steps.

1. We analyze the end-to-end delay bound of a packet given the interference of the other flows.
2. We derive an upper bound of a flow's conflict and contention interferences on a packet.
3. Combining 1) and 2), we give the upper bound of the end-to-end delay of a flow.

4. We reduce the pessimism in admission control by iteratively tightening the delay bounds of flows with short deadlines.

3.4.2 Conflict and Contention Delays

In this subsection we analyze the conflict delay and contention delay of a packet. Consider a packet $P_{k,j}$ of flow F_k released at time $r_{k,j}$ with absolute deadline $d_{k,j}$. We want to analyze the end-to-end delay of $P_{k,j}$ assuming both the conflict interferences and contention interferences of all the other flows on $P_{k,j}$ are given.

Lemma 1. *The conflict delay $Y^f_{k,j}$ of $P_{k,j}$ is upper bounded as follow:*

$$Y^f_{k,j} \leq \sum_{l \neq k} I^f_{k,j}(l). \tag{3.3}$$

Proof. For any time slot within $Y^f_{k,j}$, $P_{k,j}$ is delayed by conflict if and only if there is at least one scheduled higher priority packet conflicting with it. Let one of these higher priority packets belong to flow F_l. Recall the definition of F_l's conflict interference on $P_{k,j}$ is the number of time slots when $P_{k,j}$ is delayed due to conflicting transmissions belonging to flow F_l. By definition, this time slot is a part of F_l's conflict interference $I^f_{k,j}(l)$.

Since our statement is not limited to a specific time slot within $Y^f_{k,j}$, we show any time slot in which $P_{k,j}$ suffers one conflict delay indeed belongs to at least one flow's conflict interference on $P_{k,j}$. Therefore, the total conflict delay of $P_{k,j}$ is bounded by the sum of conflict interferences of all other flows. □

Lemma 2. *The contention delay $Y^t_{k,j}$ of $P_{k,j}$ is upper bounded as follow:*

$$Y^t_{k,j} \leq \lfloor \frac{\sum_{l \neq k} I^t_{k,j}(l)}{m} \rfloor. \tag{3.4}$$

Proof. We follow the same reasoning of the proof of the Lemma 1. For any time slot, $P_{k,j}$ is delayed by contention if and only if all channels are occupied by higher priority packets

and none of them conflict with $P_{k,j}$. Then in this time slot, there must be m higher priority packets scheduled and none of them conflict with $P_{k,j}$.

Let $I^{t'}_{k,j}(l)$ denote the number of time slots when 1) $P_{k,j}$ is ready but not executing, 2) F_l is executing and 3) none of the executing packets conflict with $P_{k,j}$. Then the contention delay of $P_{k,j}$ is upper bounded by $\lfloor \frac{\sum_{l \neq k} I^{t'}_{k,j}(l)}{m} \rfloor$. Recall that $I^{t}_{k,j}(l)$ is the number of time slots when 1) $P_{k,j}$ is ready but not executing, 2) F_l is executing and 3) F_l does not conflict with $P_{k,j}$. Comparing the set of time slots within $I^{t}_{k,j}(l)$ and $I^{t'}_{k,j}(l)$, we see the latter one is a subset of the former one, so $I^{t'}_{k,j}(l)$ is no greater than $I^{t}_{k,j}(l)$. Therefore, $Y^{t}_{k,j}$ is upper bounded by $\lfloor \frac{\sum_{l \neq k} I^{t}_{k,j}(l)}{m} \rfloor$. □

To meet $P_{k,j}$'s deadline, the end-to-end delay of $P_{k,j}$ should satisfy the following condition: $R_{k,j} = Y^{f}_{k,j} + Y^{t}_{k,j} + C_k \leq D_k$. To make flow F_k schedulable, this condition should hold for all its packets.

3.4.3 Upper Bound of Interferences

The conflict and contention delay bounds in Lemma 1 and Lemma 2 depend on the conflict and contention interferences. To give the worst-case end-to-end delay R_k of each flow F_k, the most straightforward approach is to compute every other flow F_l's conflict interference and contention interference on every packet of F_k up to the hyper-period. However, this is computationally expensive. We therefore derive upper bounds of the interferences that can be computed efficiently.

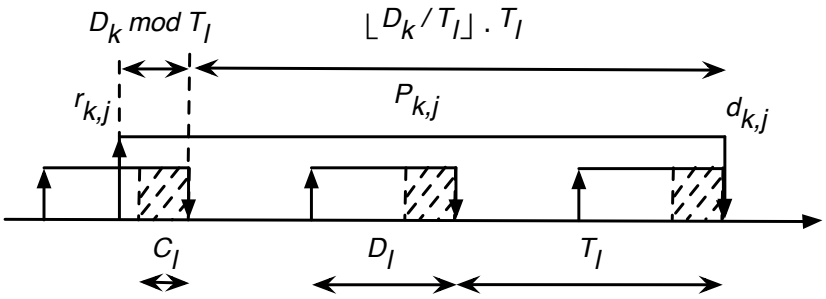


Figure 3.1: Worst-case workload of flow F_l

To start, we analyze the upper bound of the interference $I_{k,j}(l)$ which is the sum of conflict interference $I^f_{k,j}(l)$ and contention interference $I^t_{k,j}(l)$. It is obvious that the interference of flow F_l on any packet $P_{k,j}$ cannot exceed its workload within $P_{k,j}$'s lifetime $[r_{k,j}, d_{k,j}]$, where flow F_l's workload is the number of time slots when it executes. We denote the workload of F_l within $[r_{k,j}, d_{k,j}]$ as $W_{k,j}(l)$. The worst-case workload would be a upper bound of the interference. We show the worst-case workload in Figure 3.1 when the absolute deadline of one packet of F_l aligns with the absolute deadline of $P_{k,j}$. In the figure, upper arrows and down arrows represent release times and absolute deadlines of packets, respectively. $P_{k,j}$ is the j^{th} packet of flow F_k. C_l, D_l and T_l are the transmission count, the relative deadline and the period of F_l respectively. $r_{k,j}$ and $d_{k,j}$ are the release time and the absolute deadline of packet $P_{k,j}$, respectively. Dashed areas are time slots when transmissions of packets are scheduled. We give the following lemma to upper bound F_l's workload.

Lemma 3. *The workload of F_l within $[r_{k,j}, d_{k,j}]$ is upper bounded as follow:*

$$W_{k,j}(l) \leq \lfloor D_k/T_l \rfloor C_l + \min(C_l, D_k \ mod \ T_l),$$

where D_k mod T_l is the remainder of D_k divided by T_l.

Proof. We discuss the workload of F_l in three cases:

- $D_k < T_l$
- $D_k \geq T_l$ and $(D_k \ mod \ T_l) < D_l$
- $D_k > T_l$ and $(D_k \ mod \ T_l) \geq D_l$

In the first case, deadline of flow F_k is less than period of flow F_l. Within $[r_{k,j}, d_{k,j}]$, there is at most one packet of F_l active. Then the maximum workload of F_l is $\min(C_l, D_k)$, which follows this lemma.

In the second case, deadline of F_k is no less than period of F_l, and $D_k \ mod \ T_l$ is less than D_l. First, if $D_k \ mod \ T_l$ equals 0, then the number of flow F_l's packets within $[r_{k,j}, d_{k,j}]$ is D_k/T_l, and the total workload is $(D_k/T_l)C_l$, which follows this lemma. Then, we assume $D_k \ mod \ T_l > 0$, this is the exact case we show in Figure 3.1. there is one carry-in packet of flow

F_l, which is released before $P_{k,j}$ and delivered after $P_{k,j}$'s release. The number of F_l's packets within $[r_{k,j}, d_{k,j}]$ is $\lfloor D_k/T_l \rfloor$. And the carry-in packet's workload is $\min(C_l, D_k \ mod \ T_l)$. Then the total workload is $\lfloor D_k/T_l \rfloor C_l + \min(C_l, D_k \ mod \ T_l)$, which also follows this lemma.

In the third case, the number of packets of F_l that are completely contained in $[r_{k,j}, d_{k,j}]$ is $\lfloor D_k/T_l \rfloor + 1$, given that $D_k \ mod \ T_l \geq D_l$. The workload of F_l is $(\lfloor D_k/T_l \rfloor + 1)C_l$. Because $D_k \ mod \ T_l \geq D_l \geq C_l$, the workload provided by this lemma is

$$\begin{aligned} \lfloor D_k/T_l \rfloor C_l + \min(C_l, D_k \ mod \ T_l) =& \lfloor D_k/T_l \rfloor C_l + C_l \\ =& (\lfloor D_k/T_l \rfloor + 1)C_l. \end{aligned}$$

Then this case follows the lemma as well. □

We have an upper bound of F_l's interference on packet $P_{k,j}$ as:

$$I_{k,j}(l) \leq W_{k,j}(l) \leq \lfloor D_k/T_l \rfloor C_l + \min(C_l, D_k \ mod \ T_l). \tag{3.5}$$

We use $\widehat{I_{k,j}(l)}$ to denote this upper bound. Then we have

$$\widehat{I_{k,j}(l)} = \lfloor D_k/T_l \rfloor C_l + \min(C_l, D_k \ mod \ T_l). \tag{3.6}$$

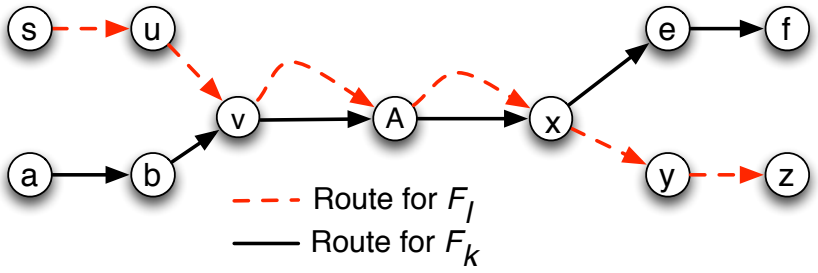


Figure 3.2: An example to show conflict delay

Now we derive the upper bound of flow F_l's conflict interference $I^f_{k,j}(l)$ on packet $P_{k,j}$. Let $S_k(l)$ denote the maximum conflict interference that one packet of flow F_l can incur on one packet of flow F_k. Packets of flows F_k and F_l conflict with each other when their transmissions

share at least one node. So $S_k(l)$ is the number of F_k's transmissions that share nodes with F_l's transmissions. It depends on the number of links in F_l's route that share nodes with F_k's route as well as the number of (re)transmissions scheduled on each link. We can count it based on the routes of the two flows.

As shown in Figure 3.2, F_k and F_l are two flows that share a part of their routes. The number of links in F_l's route that share nodes with F_k's route is 4, and they are $\{u \to v, v \to A, A \to x, x \to y\}$. For simplicity, assuming only one transmission is scheduled for each link, $S_k(l)$ in this example equals 4. After upper bounding the conflict interference that one packet of F_l can introduce, we can upper bound the total conflict interference that flow F_l can introduce on packet $P_{k,j}$.

Following the same reasoning of analyzing the maximum workload, we have the following corollary.

Corollary 1. *The conflict interference of flow F_l on packet $P_{k,j}$ is upper bounded as follow:*

$$I^f_{k,j}(l) \leq \lfloor D_k/T_l \rfloor S_k(l) + \min(S_k(l), D_k \ mod \ T_l). \tag{3.7}$$

Here we use $\widehat{I^f_{k,j}(l)}$ to denote this upper bound of $I^f_{k,j}(l)$, and

$$\widehat{I^f_{k,j}(l)} = \lfloor D_k/T_l \rfloor S_k(l) + \min(S_k(l), D_k \ mod \ T_l). \tag{3.8}$$

After upper bounding the interferences of flow F_l on packet $P_{k,j}$, we have the upper bound of end-to-end delay in the following theorem.

Theorem 1. *The end-to-end delay of flow F_k is upper bounded as follow:*

$$R_k \leq \sum_{l \neq k} \widehat{I^f_{k,j}(l)} + \lfloor \frac{\sum_{l \neq k} (\widehat{I_{k,j}(l)} - \widehat{I^f_{k,j}(l)})}{m} \rfloor + C_k = \hat{R}_k.$$

Proof. As we showed in Equation (3.1), $R_{k,j} = Y^f_{k,j} + Y^t_{k,j} + C_k$. Combining it with Lemma 1 and Lemma 2, we have following inequation:

$$R_{k,j} \leq \sum_{l \neq k} I^f_{k,j}(l) + \lfloor \frac{1}{m} \sum_{l \neq k} I^t_{k,j}(l) \rfloor + C_k. \tag{3.9}$$

From Equations (3.6) and (3.8), we have upper bounds of $I_{k,j}(l)$ and $I^f_{k,j}(l)$. However, we don't have an upper bound of $I^t_{k,j}(l)$. Based on Inequation (3.2), $I_{k,j}(l) = I^f_{k,j}(l) + I^t_{k,j}(l)$. In any case, $I^f_{k,j}(l) + I^t_{k,j}(l) \leq \widehat{I_{k,j}(l)}$, and $I^f_{k,j}(l) \leq \widehat{I^f_{k,j}(l)}$. As shown in Equation (3.9), $I^t_{k,j}(l)$ is divided by m and floored, which shows $I^f_{k,j}(l)$ has higher weight than $I^t_{k,j}(l)$ in the equation. Given the fixed upper bound of $\widehat{I_{k,j}(l)}$, any amount that we reduce from conflict interference and add to contention interference will not increase the end-to-end delay. So we use $\widehat{I_{k,j}(l)} - \widehat{I^f_{k,j}(l)}$ to replace $I^t_{k,j}(l)$, use $\widehat{I^f_{k,j}(l)}$ to replace $I^f_{k,j}(l)$ and get the upper bound of R_k as the theorem shows. The intuition is we would rather overestimate conflict interference and underestimate contention interference, which will not violate the safety of our upper bound.

□

3.4.4 Improved Delay Analysis

We give an upper bound of end-to-end delay in Theorem 1. Bertogna et al. [39] proposed a technique to iteratively improve scheduability analysis for real-time tasks under EDF scheduling. Inspired by their technique, we propose an *Improved Delay Analysis (IDA)*. From now on, we will call the end-to-end delay analysis in Theorem 1 as the *Basic Delay Analysis (BDA)* and use it as a foundation of our IDA.

We illustrate the intuition of IDA through an example shown in Figure 3.3. $\hat{R}_l$ is an upper bound of the end-to-end delay of flow F_l we obtained through BDA (Theorem 1). We consider the flow F_l's interference on packet $P_{k,j}$. In this example the deadline of packet $P_{l,h}$ is aligned with deadline of packet $P_{k,j}$. The upper bound of the end-to-end delay of F_l is shown in the figure with $\hat{R}_l$. From the figure, $P_{l,h}$ is delivered to the destination before absolute deadline of $P_{l,h}$ as well as the release time of $P_{k,j}$. Then all transmissions of $P_{l,h}$ are scheduled before the release of $P_{k,j}$. Clearly, the interference of F_l on $P_{k,j}$ is zero. However, based on BDA, the conflict interference of F_l on $P_{k,j}$ is $S_k(l)$ (maximum conflict delay that $P_{l,h}$ can incur on $P_{k,j}$), and the contention interference of F_l is not zero either. BDA hence overestimates the

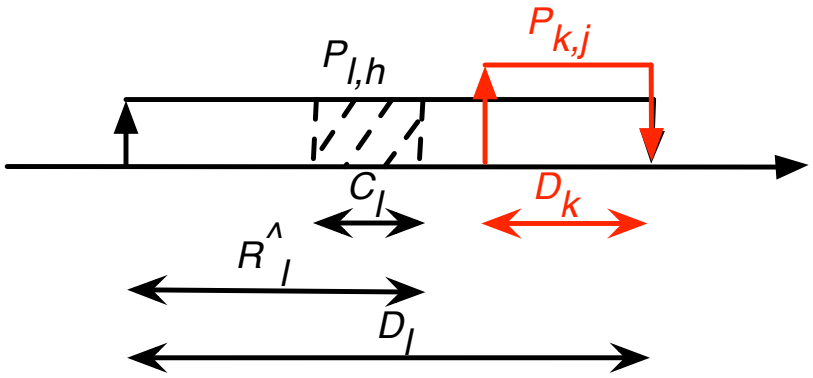


Figure 3.3: An example for Observation 1

interference in this example, because it ignores the fact that $P_{l,h}$ is delivered well before its deadline.

This observation leads to a way to reduce the pessimism of our analysis. In BDA, the most difficult part is to assure the schedulability of flows with short deadlines, because an overestimation of interference would easily push the end-to-end delay bound of the packet over its short deadline. The intuition behind IDA is to tighten up the interference estimation by considering early completion of packets.

Observation 1. *Let $\hat{R}_l$ denote an upper bound of end-to-end delay of flow F_l, no transmissions of packet $P_{l,h}$ can be scheduled later than $r_{l,h} + \hat{R}_l$.*

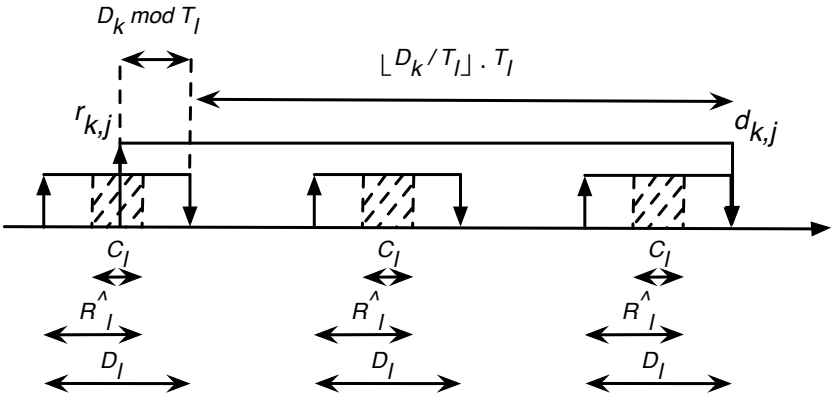


Figure 3.4: Worst-case scenario under Observation 1

By incorporating this observation, we propose our IDA. In IDA, we use superscript * to denote the new results of variables we already introduced in BDA.

We start by analyzing the upper bounds of interferences. We show the worst-case interference of F_l on packet $P_{k,j}$ in Figure 3.4. Note that we also show the the upper bound of the end-to-end delay $\hat{R}_l$ of F_l in the figure.

Lemma 4. *Flow F_l's interference on packet $P_{k,j}$ is upper bounded as follow:*

$$\widehat{I_{k,j}(l)}^* = \min(C_l, \max(0, (D_k \ mod \ T_l) - (D_l - \hat{R}_l))) + \lfloor D_k/T_l \rfloor C_l. \tag{3.10}$$

Proof. We discuss in four cases.

- $D_k < T_l$ and $D_k \leq D_l - \hat{R}_l$
- $D_k < T_l$ and $D_k > D_l - \hat{R}_l$
- $D_k \geq T_l$ and $(D_k \ mod \ T_l) \leq D_l - \hat{R}_l$
- $D_k \geq T_l$ and $(D_k \ mod \ T_l) > D_l - \hat{R}_l$

In the first case, deadline of flow F_k is less than period of flow F_l. Within $[r_{k,j}, d_{k,j}]$, there is at most one packet of F_l active. Since deadline of F_k is no greater than gap between F_l's upper bound of end-to-end delay $\hat{R}_l$ and its deadline D_l. All transmissions of F_l are scheduled before $P_{k,j}$'s release time. Then F_l's interference equals zero in this case, which follows this lemma. This case is exactly what we show in Figure 3.3.

In the second case, there is also at most one packet of F_l active. However, the release time of $P_{k,j}$ is before F_l's upper bound of end-to-end delay. Packet $P_{k,j}$ is released $D_k - (D_l - \hat{R}_l)$ time slots before one packet of F_l complete its transmissions. Then the maximum possible interference is $\min\{C_l, D_k - (D_l - \hat{R}_l)\}$, which also follows this lemma.

In the third case, deadline of F_k is no less than period of F_l, and $D_k \ mod \ T_l$ is no greater than $D_l - \hat{R}_l$. First, if $D_k \ mod \ T_l$ equals 0, then the number packets of F_l within $[r_{k,j}, d_{k,j}]$ is

D_k/T_l, and the total interference is $(D_k/T_l)C_l$, which follows this lemma. Then, we assume $0 < D_k \ mod \ T_l \leq D_l - \hat{R}_l$. There is one carry-in packet of flow F_l, which is delivered before $P_{k,j}$'s release. So its interference on packet $P_{k,j}$ is 0. Then the total interference is $\lfloor D_k/T_l \rfloor C_l$, which also follows this lemma.

In the last case, deadline of F_k is no less than period of F_l, and $D_k \ mod \ T_l$ is larger than $D_l - \hat{R}_l$. If $D_k \ mod \ T_l \geq D_l$, there are $D_k/T_l + 1$ packets of F_l contained within $[r_{k,j}, d_{k,j}]$, the total interference is $(D_k/T_l + 1)C_l$, which follows this lemma. If $D_l - \hat{R}_l < D_k \ mod \ T_l < D_l$, the carry-in packet of flow F_l is partial within $[r_{k,j}, d_{k,j}]$, which is exactly what we show in Figure 3.4. The interference of this carry-in packet depends on $D_k \ mod \ T_l - (D_l - \hat{R}_l)$, and equals $\min(C_l, (D_k \ mod \ T_l) - (D_l - \hat{R}_l)$. This case follows the lemma as well. □

Following the same reasoning of analyzing upper bound of interference, we have following corollary.

Corollary 2. *Flow F_l's conflict interference on packet $P_{k,j}$ is upper bounded as follow:*

$$\widehat{I^f_{k,j}(l)}^* = \min(S_k(l), \max(0, (D_k \ mod \ T_l) - (D_l - \hat{R}_l))) + \lfloor D_k/T_l \rfloor S_k(l). \tag{3.11}$$

Similar to Theorem 1, we give an upper bound of end-to-end delay of F_k here.

Corollary 3. *The worst-case end-to-end delay of flow F_k is upper bounded as follow:*

$$R_k \leq \sum_{l \neq k} \widehat{I^f_{k,j}(l)}^* + \lfloor \frac{\sum_{l \neq k} (\widehat{I_{k,j}(l)}^* - \widehat{I^f_{k,j}(l)}^*)}{m} \rfloor + C_k = \hat{R_k}^*. \tag{3.12}$$

The flow set $\{F_1, F_2, \cdots, F_n\}$ is schedulable if the following statement is true:

$$\hat{R_k}^* \leq D_k, \qquad k = 1, 2, \cdots, n. \tag{3.13}$$

We use an iterative algorithm to derive the upper bound of end-to-end delay $\hat{R_k}^*$. In the beginning, the initial upper bound $\hat{R_k}$ is set to D_k for all flows. In each iteration, $\hat{R_k}^*$ is

calculated based on Equation (3.10)-(3.12). At the end of each iteration, for each flow F_k, $\hat{R}_k$ is set to $\hat{R}_k^*$. The algorithm enters a new iteration if the flow set is unschedulable and at least one flow has $\hat{R}_k^*$ updated, otherwise it terminates. We show the pseudo-code in Algorithm 1.

Algorithm 1: Iterative algorithm

$\hat{R}_k \leftarrow D_k, \forall k \leq N$;
repeat
  for $k \leq N$ **do**
    $\hat{R}_k^0 \leftarrow \hat{R}_k$;
    Calculate $\hat{R}_k^*$ based on (3.10)-(3.12);
    $\hat{R}_k \leftarrow \hat{R}_k^*$;
  end
until $\hat{R}_k \leq D_k$ ***or*** $\hat{R}_k = \hat{R}_k^0, \forall k \leq N$;
$\hat{R}_k^* \leftarrow \hat{R}_k, \forall k \leq N$;

3.4.5 Complexity Analysis

BDA (Theorem 1) is polynomial. The calculation of upper bound of end-to-end delay of flow F_k is $O(n)$ since we have n flows. The complexity of BDA is $O(n^2)$ since we need to calculate the upper bound of end-to-end delay for every flow. The total time complexity is therefore $O(n^2)$.

IDA (Corollary 3) is pseudo-polynomial. The analysis in each iteration is $O(n^2)$ as discussed above. Since there are n flows, and each one's end-to-end delay can range from C_k to D_k, the number of iterations is upper bounded as $O(n \max(D_k - C_k, k \leq N))$. Thus, the overall complexity is $O(n^3 \max(D_k - C_k, k \leq N))$.

3.5 Evaluation

We evaluate our end-to-end delay analysis through both experiments on a physical WSAN testbed and simulations.

3.5.1 Experiments on a WSAN Testbed

We evaluate our delay analysis on an indoor WSAN testbed consisting of 63 TelosB motes, located on the fifth floors of Bryan Hall and Jolly Hall of Washington University in St. Louis. We implement a network protocol stack on the testbed, which comprises a multi-channel TDMA MAC protocol and a routing protocol. Time is divided into 10 ms slots and clocks are synchronized across the entire network using the Flooding Time Synchronization Protocol (FTSP) [143]. In the routing protocol, we want to find the maximum number of link-disjoint paths for any pair of nodes. We transform this problem into a maximum flow problem by assigning each link with unit capacity and use Edmonds−Karp algorithm [71] to generate link-disjoint routes.

Figure 3.5 shows the topology of the WSAN testbed. We use motes 129 and 155 (red circles in Figure 3.5) as access points, which are physically connected to a root server (Gateway). The other motes are used as field devices (red circles in Figure 3.5). Black arrows are wireless links. The Network Manager runs on this root server. The rest of motes work as field devices. For each link in the testbed, we measured its *packet reception ratio (PRR)* by counting the number of received packets among 250 packets transmitted on the link. Following the practice of industrial deployment, we only add links with PRR higher than 90% to the topology of the testbed. To avoid channels occupied by the campus Wi-Fi, we use IEEE 15.4 channel 11 to 15 in our experiments.

We generate 8 flows in our experiment. The period of each flow is picked up from the range of $2^{0\sim7}$ seconds, which are typical periods used in process industry as defined in WirelessHART standard [26]. The length of the hyper-period is 128 seconds. The relative deadline of each flow equals to its period. All flows are schedulable based on our delay analyses. Each flow has two independent source routes. The maximum length of routes is 13 hops. Through this double-route approach, we enhance the network reliability under link failures. We run our experiments long enough such that each flow can deliver at least 100 packets.

Based on our experimental results, we evaluate our proposed approaches in terms of reliability and delay. We use *delivery ratio* to measure reliability. The *delivery ratio* of a flow is defined as percentage of packets that are successfully delivered to destination. Then, we compare the

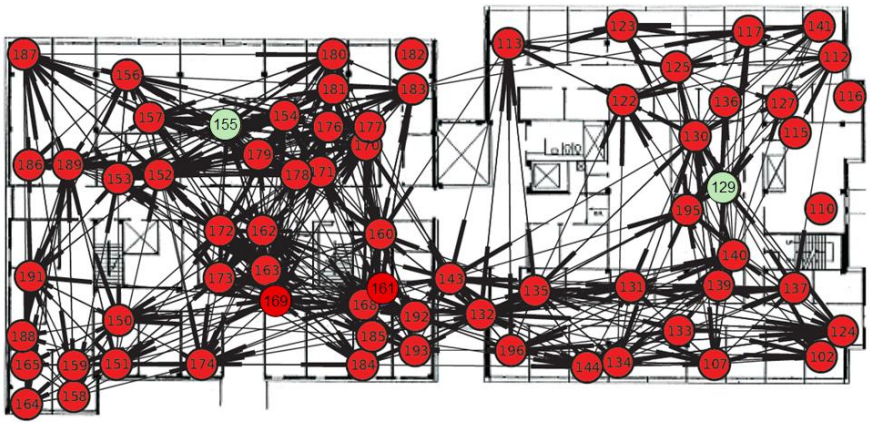


Figure 3.5: Topology of the WSAN Testbed

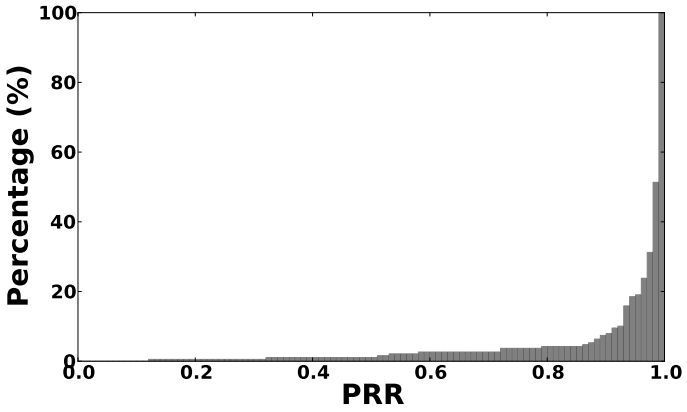


Figure 3.6: Cumulative Histogram of Link Qualities

Flow Index	1	2	3	4	5	6	7	8
1^{st} Route	0.95	1.0	0.97	1.0	0.97	0.96	0.97	0.97
2^{nd} Route	1.0	1.0	0.99	0.99	1.0	0.97	1.0	0.42
Two Routes	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.99

Table 3.1: Delivery Ratios of Flows

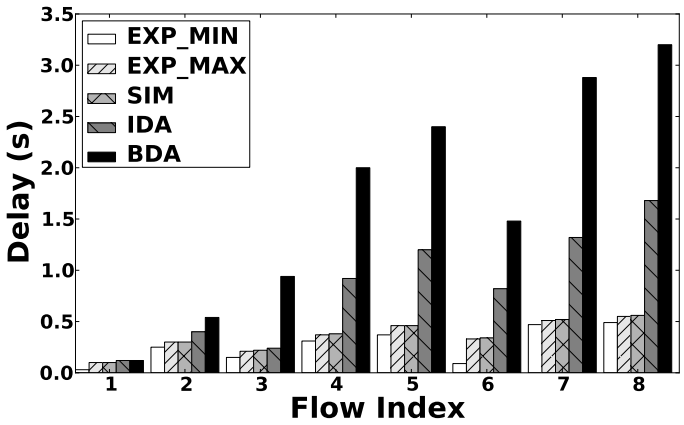


Figure 3.7: End-to-End Delays

end-to-end delay we collected in experiments with our delay analyses, as well as the delay observed in simulations.

To study the reliability issue, we first measure the link qualities in our testbed . Figure 3.6 shows the cumulative histogram of link qualities (PRR) of 189 links we used in our experiments. Although we only picked up links that have PRR higher than 90% when we selected the links initially, we find some links have much lower PRR than the 90% threshold at run time. For example, link 112 $\rightarrow$ 129 has the lowest PRR of 12%. The dynamics of wireless links suggest it is necessary to have route redundancy.

Table 3.1 shows the delivery ratios of all 8 flows. We present the delivery ratio of each route as well as the aggregate delivery ratio of the two routes combined. Our results demonstrate the effectiveness of redundant routes in improving reliability. For example, the second route of flow 8 has a delivery ratio of 0.42, which is much lower than our expectation. However, by combing two routes together, flow 8 has a delivery ratio as 0.99.

In Figure 3.7, we compare end-to-end delay from experiment results with delay analyses as well as simulation. We compare five delays for each flow: minimum delay in experiments (EXP_MIN), maximum delay in experiments (EXP_MAX), maximum delay in simulation (SIM), the improved delay analysis (IDA) in Corollary 3 and the basic delay analysis (BDA) in Theorem 1. The results show for every flow, the five delays follow the following order:

EXP_MIN $\leq$ EXP_MAX $\leq$ SIM $\leq$ IDA $\leq$ BDA.

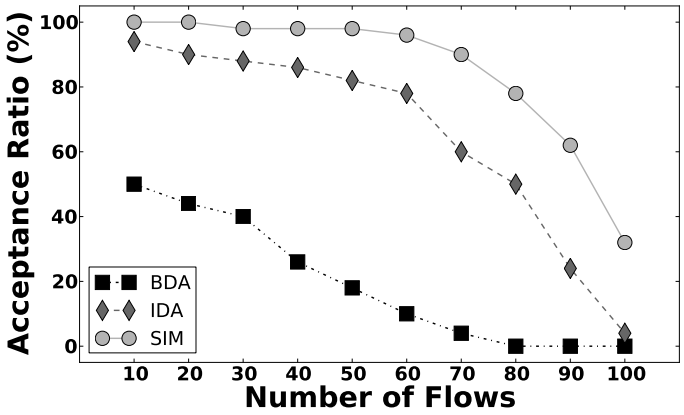
This shows that our delay analyses are safe upper bounds of the actual delays. In addition, SIM is consistently higher than EXP_MAX, which indicates our simulations can generate test cases with worse delays than those observed on the testbed. In following evaluation, we will provide a more comprehensive evaluation of the delay analyses based on simulations over different network topologies.

3.5.2 Simulations on Random Topologies

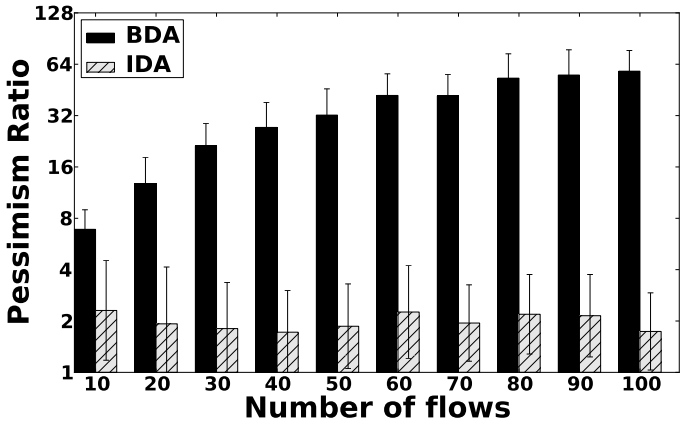
Besides the Testbed experiments, we also test our analyses on larger random topologies with simulations. The simulator shares the same routing and scheduling design with our testbed and is written in C++. All simulations are performed on a MacBook Pro laptop with 2.4 GHz Intel Core 2 Duo processor. We generate random networks with 400 nodes and 800 links. Links are chosen randomly and assigned PRR randomly in the range of $[0.90, 1.0]$. We test our delay analyses on different number of flows by increasing the number of source and destination pairs. The period T_k of the each flow F_k is randomly generated in the range of $2^{3\sim9}$ seconds. The relative deadline D_k of every flow F_k is randomly generated in the range of $(C_k, \beta * T_k)$ slots, here β is a randomly generated number in range of $(0, 1)$. C_k is the required time slots needed to deliver a packet from the source to the destination. For each flow set, we generate 100 test cases and simulate them on random topologies.

We compare our improved delay analysis (IDA) in Corollary 3 with the basic delay analysis (BDA) in Theorem 1 and the simulation (SIM). Our delay analyses are evaluated in terms of *pessimism ratio* and *acceptance ratio*. The former one is used to assess the tightness of the delay analyses, and the latter one is used to evaluate the effectiveness of our analyses for online admission control. For each flow, the *pessimism ratio* is defined as the ratio of its theoretical upper bound of end-to-end delay given by our analyses to its maximum end-to-end delay observed in simulation. The *acceptance ratio* is defined as the ratio of the number of test cases deemed schedulable by our analyses (or simulation) to the total number of test cases. A test case is schedulable in simulation if all flow instances released within the hyper-period meet their deadlines.

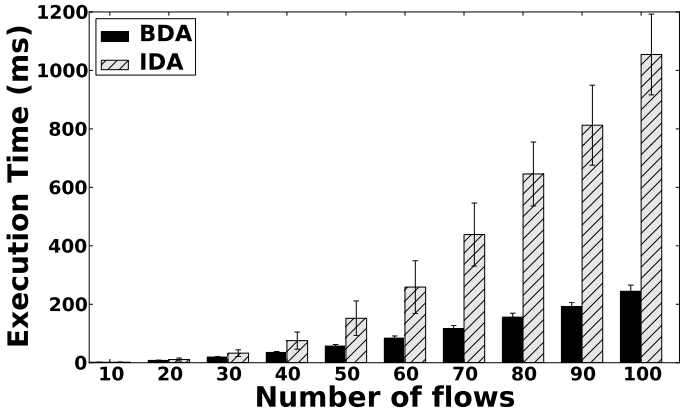
The acceptance ratios of IDA, BDA and simulation (SIM) are shown in Figure 3.8(a). The acceptance ratio of IDA remains close to simulations. The gap between IDA and SIM



(a) Acceptance Ratio



(b) Pessimism Ratio (in log scale)



(c) Execution Time

Figure 3.8: Schedulability Analysis on Random Topology

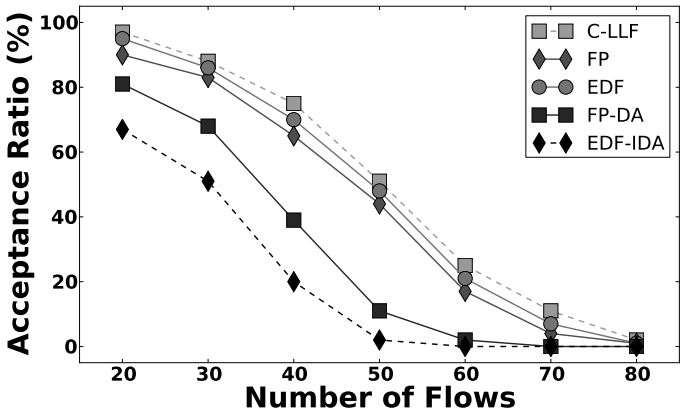
widens as the number of flows increases, but remains within 30%. This result indicates the effectiveness of IDA for admission control. The acceptance ratio of IDA is much higher than BDA, which shows the IDA highly outperforms BDA in terms of acceptance ratio.

Figure 3.8(b) shows pessimism ratios of IDA and BDA in log scale. Since if a test case is not schedulable under simulation, the simulator could not lay out the schedule of all flows, then we could not get the actual maximum end-to-end delay. So all pessimism ratios here are from test cases that are schedulable under simulation. This result confirms IDA greatly improves the tightness of the delay bound compared to BDA. The pessimism ratio of BDA increases as the number of flows increases. However, the pessimism ratio of IDA remain low despite the increase of number of flows. The median value of pessimism ratio for IDA is always around 2 in our simulations. This figure shows our IDA is scalable to large number of flows.

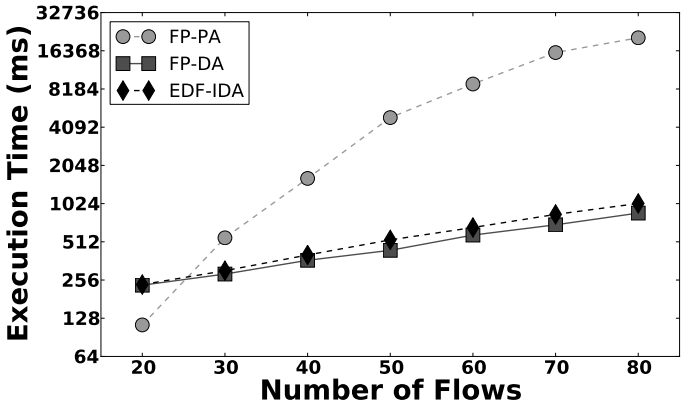
The time complexity of our algorithms are shown in Figure 3.8(c). The execution time of IDA grows faster than BDA as the number of flows grows while staying in an acceptable region. With 100 flows, the execution time of IDA is under 1.2 seconds, which is acceptable for admission control. Figures 3.8(a)-3.8(c) show the tradeoff between accuracy and time complexity. While IDA runs slower than BDA, it gives a much more precise estimation of end-to-end delay, which leads to a higher acceptance ratio.

3.5.3 Comparative Study of Scheduling Policies

In this subsection, we compare EDF and our Improved Delay Analysis (EDF-IDA) with state-of-the-art dynamic and static priority scheduling algorithms and their delay analyses. For dynamic priority scheduling, we consider Conflict-aware Least Laxity First (C-LLF) [172], which incorporates transmission conflicts into a Least Laxity First scheduling policy. However, there is no delay analysis for C-LLF in the literature. For fixed priority scheduling, we choose Fixed Priority with near optimal priority assignment based on heuristic search (FP) presented in [171], which was shown to significantly outperform traditional priority assignment policies. Delay Analysis for Fixed Priority scheduling policies (FP-DA) has been proposed in [170].



(a) Acceptance Ratio



(b) Execution Time (in log scale)

Figure 3.9: Comparison of Different Scheduling Policies

We compare scheduling policies through simulations on random topologies. To fully test the schedulability of different scheduling algorithms, we reduce the periods of flows from the range of $2^{6\sim11}$ to $2^{5\sim10}$. The rest of the simulation setups are same as the previous subsection. Figure 3.9(a) shows the acceptance ratios of different scheduling policies and their delay analyses. Since these is no delay analysis for C-LLF, we only include its acceptance ratio in simulations. Results show in simulations, EDF can schedule more flow sets than FP, which indicates that EDF is indeed an effective scheduling policy in practice. While C-LLF can schedule more flow sets than EDF, there is no schedulability analysis for C-LLF that can be used for fast online admission control. We also compare the acceptance ratios of delay analyses EDF-IDA and FP-DA here. Given the complexity that EDF brings to schedulability analysis, the acceptance ratio of EDF-IDA is slightly lower than FP-DA.

Figure 3.9(b) shows execution time (in log scale) of the delay analyses as well as the priority assignment algorithm needed by FP (denoted as FP-PA). The execution time of FP-PA is much higher than execution time of EDF-IDA and FP-DA. Note that priority assignment is an integral part of fixed priority scheduling and the near optimal priority assignment needs to be performed for admission control of flows. Given the high computational cost of priority assignment algorithm, EDF-IDA provides a more efficient admission test for real-time flows, which is particularly important for WSANs operating under dynamic wireless conditions in industrial environments.

3.6 Summary

With the emergence of industrial standards such as WirelessHART, wireless sensor-actuator networks (WSANs) are gaining rapid adoption in process industries. To meet the stringent real-time performance requirements of process control systems, there is a critical need for fast end-to-end delay analysis to support online admission control of periodic real-time flows in WSANs. This chapter presents a new end-to-end delay analysis for WSANs under Earliest Deadline First (EDF) transmission scheduling, a widely used dynamic priority scheduling policy in real-time systems. Our analysis that can be used to derive end-to-end delay bounds for real-time flows in WSANs at moderate run time overhead. Experiments

on a physical WSAN testbed and simulations demonstrate the effectiveness of our analysis for online admission control of real-time flows.

Chapter 4

Real-Time Routing for Wireless Sensor-Actuator Networks

As process industries start to adopt wireless sensor-actuator networks (WSANs) for control applications, it is crucial to achieve real-time communication in this emerging class of networks. Routing has significant impacts on end-to-end communication delays in WSANs. However, despite considerable research on real-time transmission scheduling and delay analysis for such networks, real-time routing remains an open question for WSANs. This chapter presents a *conflict-ware real-time routing* approach for WSANs. This approach leverage a key observation that conflicts among transmissions sharing a common field device contribute significantly to communication delays in industrial WSANs such as WirelessHART networks. By incorporating conflict delays in the routing decisions, conflict-aware real-time routing algorithms allow a WSAN to accommodate more real-time flows while meeting their deadlines. Evaluation based on simulations and experiments on a real WSANs testbed show conflict-aware real-time routing can lead to up to three-fold improvement in real-time capacity of WSANs.

4.1 Introduction

With the emergence of industrial standards such as WirelessHART [26] and ISA100.11a [16], process industries are adopting *Wireless Sensor-Actuator Networks (WSANs)* that enable sensors and actuators to communicate through low-power wireless mesh networks [184]. In recent years, we have seen world-wide deployment of WSANs. Technical reports [13] from the

process industry show more than 1900 WirelessHART networks have been deployed around the world, with more than 3 billion operating hours in the field.

Feedback control loops in industrial environments impose stringent end-to-end delay requirements on data communication. To support a feedback control loop, the network periodically delivers data from sensors to a controller and then delivers control commands to the actuators within an end-to-end deadline. The effects of deadline misses in data communication may range from production inefficiency, equipment destruction to irreparable financial and environmental damages.

Previous works [170,172,200] demonstrate that the end-to-end delays of flows highly depend on routes. It is important to optimize routes to improve the real-time capacity of WSANs. Existing routing algorithms usually select routes with the minimum hop count, which introduces high transmission conflicts among different flows. Since high transmission conflicts cause long end-to-end delays, shortest paths usually lead to a low real-time capacity. This paper presents our real-time routing algorithms for WSANs. We incorporate conflict delays into our routing design and propose conflict-aware routing algorithms that allow WSANs to accommodate more real-time flows. Our conflict-aware routing algorithms reduce conflict delays of real-time flows so they can meet their deadline constraints. Our evaluation shows that our real-time routing algorithms can greatly improve the real-time capacity of the network.

The rest of the chapter is organized as follows. Section 4.2 reviews the related works. Section 4.3 discusses the problem formulation. Section 4.4 provides a brief review of the existing delay analyses, and Section 4.5 presents our real-time routing algorithms. Section 4.6 evaluates our routing algorithms through experiments and simulations, then Section 4.7 concludes the chapter.

4.2 Related Work

WSANs have attracted much attention in the research community [94,129,167,170,172,200] recently. Previous works studied real-time transmission scheduling [61,130,172], communication delay analysis [170,200] and rate selection [167,168]. All these works assume the routes

of the flows are given, and do not provide any routing protocol. There has been increasing interest in developing routing algorithms for WSANs. For example, Han et al. [94] propose routing algorithms to build reliable routes based on hop count, but their algorithms do not consider real-time performance.

Real-time routing has been studied in the wireless sensor network community. Xu et al. [207] propose a Potential-based Real-Time Routing (PRTR) protocol that minimizes delay for real-time traffic. However, their end-to-end delay bounds are probabilistic based on network calculus theory, which is not applicable to WSANs that require strict delay bounds. SPEED [98] bounds the end-to-end communication delays by enforcing a uniform delivery velocity. MM-SPEED [74] extends SPEED to support different delivery velocities and levels of reliability. RPAR [58] achieves application-specified communication delays at low energy cost by dynamically adapting transmission power and routing decisions. However, SPEED, MM-SPEED, and RPAR all assume each device knows its location via GPS or other localization services, which is not always feasible in WSANs. Moreover, the stateless routing policies adopted by these algorithms can not provide end-to-end delay bounds. Despite existing results on the general problem of real-time routing, none of the aforementioned work can be applied to WSANs. To meet this open challenge in industrial WSANs, we investigate the problem of real-time routing in WSANs in this paper.

4.3 Problem Formulation

In this section, we discuss the problem formulation. We consider a WSAN with a set of real-time flows $\mathcal{F} = \{F_1, F_2, \cdots, F_n\}$. Each flow $F_k = (s_k, d_k, \phi_k, D_k, T_k)$ is characterized by a source s_k, a destination d_k, a source route ϕ_k[1], a relative deadline D_k, and a period T_k.

We assume that all flows are ordered by priorities. Flow F_i has a higher priority than flow F_j, if and only if $i < j$. In practice, priorities are assigned based on deadlines, periods, or

[1]The WirelessHART standard supports two types of routing: source routing and graph routing. Source routing provides a single route for each flow, whereas graph routing provides multiple redundant routes in a routing graph and therefore enhances reliability through route diversity. Our routing algorithms currently assume source routing and can be easily extended to a model where each flow has multiple source routes and sends redundant packets through every route to enhance reliability. Supporting graph routing is part of our future work.

the criticality of the real-time flows. In this work, we use the deadline-monotonic priority assignment policy [**?**], where flows with shorter deadlines are assigned with higher priorities.

Under a fixed priority scheduling policy, the transmissions of the flows are scheduled in the following way. Starting from the highest priority flow F_1, the following procedure is repeated for every flow F_i in decreasing order of priority. For the current priority flow F_i, the network manager schedules its transmissions along its route (starting from the source) in the earliest available time slots and on available channels. A time slot is available if no conflicting transmission is already scheduled in that slot. In a WSAN, the complete schedule is divided into superframes. A superframe consists of transmissions in a series of time slots and represents the communication pattern of a group of devices. A superframe repeats itself when it completes all its transmissions.

The goal of our routing algorithm is to find routes for the flows so that every flow can meet its deadline. Shortest path algorithms based on hop count [94] are commonly adopted in practice in WSANs. However, as shown in our simulation results presented in this paper, the effectiveness of these algorithms is far from the optimal. Based on the insights from end-to-end delay analyses, we propose two heuristics to assign routes to meet real-time requirements.

4.4 Conflict Delay Analysis

In this section, we summarize the delay analysis for WSANs. Previous works have studied end-to-end communication delays in WSANs [170, 200]. Based on their analyses, a packet can be delayed for two reasons: conflict delay and contention delay. Due to the half-duplex radio, two transmissions conflict with each other if they share a node (sender or receiver). In this case, only one of them can be scheduled in the current time slot. Therefore, if a packet conflicts with another packet that has already been scheduled in the current time slot, it has to be postponed to a later time slot, resulting in *conflict delay*. As a WSAN does not allow concurrent transmissions in the same channel, each channel can accommodate only one transmission across the network in each time slot. If all channels are assigned to transmissions of other packets, a packet must be delayed to a later slot, resulting in *contention delay*.

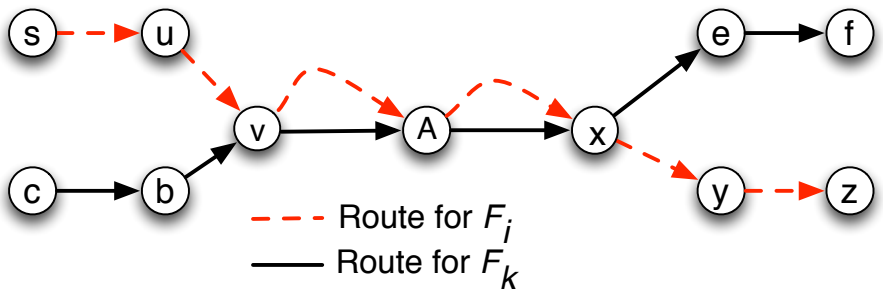


Figure 4.1: An example showing conflict delay

From existing delay analyses [170, 200] as well as our simulations, conflict delay plays a significant role in the end-to-end delays of flows. Furthermore routing directly impacts conflict delays, whereas contention delays largely depend on the number of channels available. Therefore, in our routing design, we focus only on conflict delay. Saifullah et al. proposed the Efficient Delay Analysis algorithm (EDA) in [170]. Here we briefly discuss their EDA algorithm and our approximation of EDA for our routing design.

We denote the maximum conflict delay that a package of flow F_k suffers from a package of flow F_i as Δ_k^i. Δ_k^i is counted based on the routes of the two flows. Δ_k^i equals the number of links in F_i's route that share nodes with F_k's route, times the number of transmissions scheduled on each link. We use κ to denote the number of transmissions scheduled for each link. We use an example in Figure 4.1 to show how to count Δ_k^i. F_k and F_i are two flows that share a part of their routes. Four links in F_i's route share nodes with F_k's route, which are $\{(u,v),(v,A),(A,x),(x,y)\}$. For simplicity, assuming only one transmission is scheduled for each link, Δ_k^i in this example equals 4.

Given a time interval of t slots, the number of packets of flow F_i that contribute to the delay of a packet of flow F_k during this time interval is upper bounded by $\lceil \frac{t}{T_i} \rceil$. As [170] shows, the worst-case conflict delay of a packet of flow F_k from all packets of flow F_i in a time window t can be bounded as

$$\Theta_k^i(t) = \lceil \frac{t}{T_i} \rceil \Delta_k^i, \tag{4.1}$$

where T_i is the period of flow F_i and Δ_k^i is the maximum conflict delay imposed by one packet of flow F_i.

By summarizing conflict delays from all flows with higher priorities than flow F_k, EDA proposes a upper bound of the conflict delay of flow F_k as

$$\Theta_k(t) = \sum_{i<k} \lceil \frac{t}{T_i} \rceil \Delta_k^i. \tag{4.2}$$

Based on Equation 4.2, EDA uses an iterative fixed-point algorithm to get the upper bound of F_k's conflict delay. However, the iterative fixed-point algorithm is too expensive for our routing algorithms since we will use the delay analysis as a basic component and call it extensively in our routing algorithm. Here, we propose an efficient approximation of EDA.

A packet of flow F_k can be delayed only within its lifetime D_k (the relative deadline of flow F_k). Instead of using an iterative fixed-point algorithm, we use the deadline of flow F_k as the length of time window. We further ignore the ceiling function and approximate the conflict delay that F_k can suffer from flow F_i as

$$\Theta_k^i = \frac{D_k}{T_i} \Delta_k^i. \tag{4.3}$$

By considering conflict delays from all flows, we approximate the conflict delay of flow F_k as:

$$\Theta_k = \sum_{i<k} \frac{D_k}{T_i} \Delta_k^i. \tag{4.4}$$

We present the pseudocode of our conflict delay analysis algorithm in Algorithm 2. The for loop from line 7 to line 11 has a complexity of $O(|\phi_i| log |\phi_k|)$ since one look up takes $log|\phi_k|$ in average. The for loop from line 2 to line 11 has a complexity of $|\mathcal{F}||\phi_i| log |\phi_k|$. Because $|\phi_i| \leq |V|$ and $|\phi_k| \leq |V|$, the complexity of our conflict delay analysis algorithm is $O(|\mathcal{F}||V| log |V|)$.

Algorithm 2: Conflict Delay Analysis

Function *CDA*$(G, \mathcal{F}, \kappa)$

Input : A graph $G(V, E)$, a flow set $\mathcal{F} = \{F_1, F_2, \cdots, F_n\}$ ordered by priority, where $F_k = (s_k, d_k, \phi_k, T_k, D_k)$

Output : Conflict delays $\{\theta_1, \theta_2, \cdots, \theta_n\}$ for all flows

for ***each*** *flow* F_k ***from*** F_2 ***to*** F_n **do**
- $S = \emptyset$;
- **for** ***each*** *link* $(u, v) \in \phi_k$ **do**
  - insert u into S;
  - insert v into S;
- **for** ***each*** *flow* F_i ***from*** F_1 ***to*** F_{k-1} **do**
  - $\Delta_k^i = 0$;
  - **for** ***each*** *link* $(u, v) \in \phi_i$ **do**
    - **if** $u \in S$ ***or*** $v \in S$ **then**
      - $\Delta_k^i = \Delta_k^i + \kappa$;

for ***each*** *flow* F_k ***from*** F_1 ***to*** F_n **do**
- $\Theta_k = 0$;
- **if** $k > 1$ **then**
  - **for** ***each*** *flow* F_i ***from*** F_1 ***to*** F_{k-1} **do**
    - $\Theta_k = \Theta_k + \frac{D_k}{T_i}\Delta_k^i$;

4.5 Real-Time Routing

In WSANs, existing routing algorithms [94] usually take hop count as the metric when selecting routes. As a result, each flow will select a route with the minimum hop count. However, the shortest path does not necessarily lead to the smallest end-to-end delay. As previous delay analyses [170, 200] and our simulations presented in Section 4.6 show, conflict delay plays an important role in the end-to-end delay. In this section, we take conflict delay into account in the routing decision and propose our real-time routing algorithms.

As we summarized in Section 4.4, the conflict delay that a flow F_k experiences is approximated as $\Theta_k = \sum_{i<k} \frac{D_k}{T_i} \Delta_k^i$, where T_i is the period of a high-priority flow F_i, and Δ_k^i is the maximum conflict delay imposed by one packet of flow F_i. To be more specific, Δ_k^i is the number of transmissions of flow F_i that share nodes with flow F_k, which depends on the routes of flows F_i and F_k. In our real-time routing algorithms, we aim to reduce the conflict delay caused by high-priority flows under a deadline-monotonic priority assignment that assigns higher priorities to flows with shorter deadlines. This policy can improve the number of flows meeting their deadlines, as shown in our simulation results in Section 4.6.

4.5.1 Conflict-Aware Routing

We discuss our Conflict-Aware Routing (CAR) algorithm, which pick routes with small conflict delays caused by high-priority flows. Our CAR algorithm runs as follows. We assign routes for flows following the priority order, from the highest to the lowest. For each flow F_k, we update the link weights based on routes of higher priority flows. If a link (u, v) shares at least one node with a higher priority flow F_i's route, its weight will be increased by $\frac{D_k}{T_i}$ based on Equation (4.3). After updating the link weights, we run Dijkstra's algorithm [65] to find the path ϕ_k with the smallest path weight. The algorithm terminates when the flow with lowest priority is assigned with a route ϕ_n. We present the pseudocode of our CAR algorithm in Algorithm 3.

Figure 4.2 shows an example of our CAR algorithm. In this example, we have two flows, F_h and F_l. Flow F_h has a higher priority than flow F_l. The flow F_h has a source p, a destination a, a period $1s$, and a deadline $1s$. The flow F_l has a source q, a destination a, a period

Algorithm 3: Conflict-Aware Routing

Function *CAR*$(G, \mathcal{F})$

Input : A graph $G(V, E)$, A flow set $\mathcal{F} = \{F_1, F_2, \cdots, F_n\}$ ordered by priority with $F_k = (s_k, d_k, T_k, D_k)$

Variable: link weight w, link delay coefficient c

Output : A route ϕ_k for each flow F_k

for *each* *link* $(u, v) \in E$ **do**

  $w_{(u,v)} = 1$;

  $c_{(u,v)} = 0$;

for *each* *flow* F_k ***from*** F_1 ***to*** F_n **do**

  if $k > 1$ **then**

    for *each* *link* $(u, v) \in E$ **do**

      $w_{(u,v)} = 1 + D_k \cdot c_{(u,v)}$;

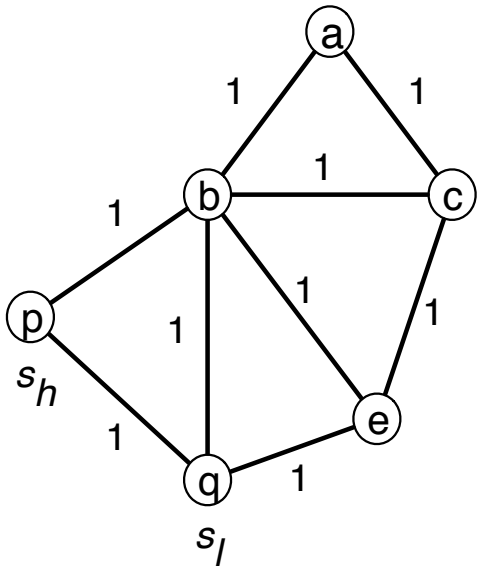
  Find the shortest path ϕ_k connecting s_k to d_k;

  Assign ϕ_k as flow F_k's route;

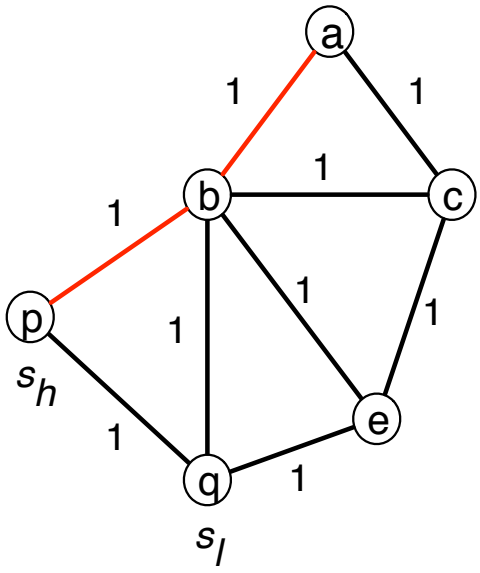
  for *each* *link* $(u, v) \in E$ **do**

    if (u, v) *shares at least one node with* F_k*'s route* R_k **then**

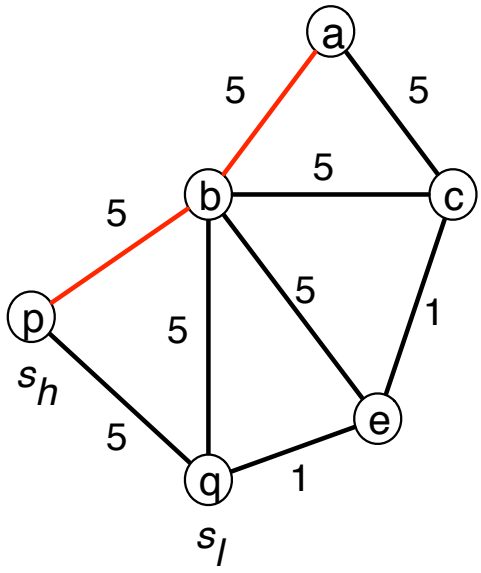
      $c_{(u,v)} = c_{(u,v)} + \frac{1}{T_k}$;



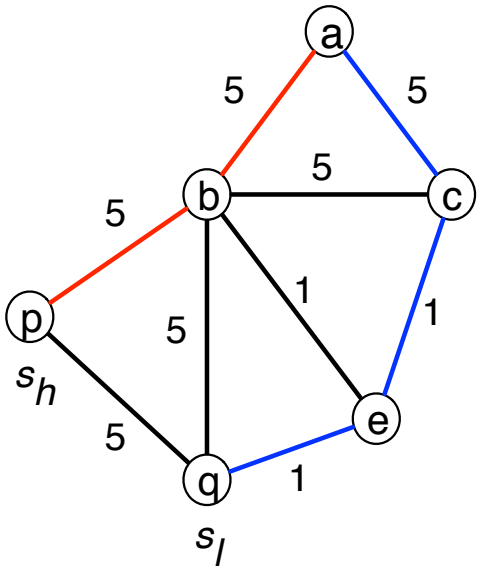
(a) Assign initial link weights



(b) Pick the route of F_h



(c) Update link weights



(d) Pick the route of F_l

Figure 4.2: An example of the CAR algorithm. Red lines represent the route of flow F_h. Blue lines represent the route of flow F_l.

$4s$, and a deadline $4s$. We use black lines to represent links in the network, red lines to represent the route of flow F_h, and blue lines to represent the route of flow F_l. In the first step (Figure 4.2(a)), we assign an initial link weight of 1 for each link in the topology. In the second step (Figure 4.2(b)), we run the shortest path algorithm to get F_h's route as $p \rightarrow b \rightarrow a$. In the second step (Figure 4.2(c)), we update the link weights based on flow F_h's route. If a link (u, v) shares at least one node with any link on flow F_h's route, we add an estimated conflict delay $\frac{D_l}{T_h} = 4$ to the link weight, because each link in flow F_h's route will bring $\frac{D_l}{T_h} = 4$ conflict delay to flow F_l based on the delay analysis in Equation 4.3. In this example, links that could encounter conflict delay from flow F_h will have a link weight of 5. In the fourth step (Figure 4.2(d)), we find the shortest path from flow F_l's source q to its destination a, which is $q \rightarrow e \rightarrow c \rightarrow a$ in this example. Note the path we found is different from the shortest path based on hop count $q \rightarrow b \rightarrow a$.

Now we discuss the complexity of the CAR algorithm. We first check the complexity for each flow (one iteration within the for loop at lines 5-13). The complexity to update the link weights is $O(|E|)$. The complexity of the Dijkstra's algorithm is $O(|E| + |V|log|V|)$, and the complexity to update the delay coefficients is $O(|E|)$. Then the complexity of each flow is $O(|E| + |V|log|V|)$. Therefore, the complexity of our CAR algorithm is $O(|\mathcal{F}|(|E| + |V|log|V|))$.

4.5.2 Iterative Conflict-Aware Routing

By reducing the conflict delay of low priority flows, we can accommodate more flows while meeting their deadlines. However, CAR is based on flow priorities, and high priority flows are not aware of the routes of low priority flows. We further improve the real-time capacity by introducing an approach where high priority flows also take into account the routes of low priority flows. We introduce our Iterative Conflict-Aware Routing (ICAR) algorithm as Algorithm 4.

The ICAR algorithm terminates when no flows update their routes in the last round or all flows are schedulable under EDA. Within each round, flows pick their routes one by one. For each flow F_k, the algorithm first updates link weights based on the routes of other flows. One difference between ICAR and CAR is that lower priority flows can also contribute to

Algorithm 4: Iterative Conflict-Aware Routing

Function *ICAR*$(G, \mathcal{F})$

Input : A graph $G(V, E)$, A flow set $\mathcal{F} = \{F_1, F_2, \cdots, F_n\}$ ordered by priority with $F_k = (s_k, d_k, T_k, D_k)$

Variable: link weight w, per link flow set S, link delay coefficient c

Output : A route ϕ_k for each flow F_k

$changed = true$; $schedulable = false$;

for ***each*** *flow* $F_k \in \mathcal{F}$ **do**

  $\phi_k = \emptyset$;

for ***each*** *link* $(u, v) \in E$ **do**

  $S_{(u,v)} = \emptyset$; $c_{(u,v)} = 0$;

while $changed == true$ ***and*** $schedulable == false$ **do**

  $changed = false$; $schedulable = true$;

  for ***each*** *flow* F_k ***from*** F_1 ***to*** F_n **do**

    if $k > 1$ **then**

      for ***each*** *link* $(u, v) \in E$ **do**

        if $F_k \in S_{(u,v)}$ **then**

          $w_{(u,v)} = 1 + D_k \cdot (c_{(u,v)} - \frac{1}{T_k})$;

        else

          $w_{(u,v)} = 1 + D_k \cdot c_{(u,v)}$;

    Find the shortest path ϕ_{temp} connecting s_k to d_k;

    $schedulable_{temp} = EDA(\phi_{temp})$;

    if $\phi_k == \emptyset$ ***or*** $(\phi_{temp} \neq \phi_k$ ***and*** $schedulable_{temp} == true)$ **then**

      $routechanged = true$; $schedulable = schedulable_{temp}$;

    if $\phi_{temp} == \phi_k$ ***or*** $(\phi_{temp} \neq \phi_k$ ***and*** $schedulable_{temp} == false)$ **then**

      $routechanged = false$; $schedulable = EDA(\phi_k)$;

    if $routechanged == true$ **then**

      $changed = true$;

      for ***each*** *link* $(u, v) \in \phi_k$ **do**

        if $(u, v) \notin \phi_{temp}$ **then**

          Remove F_k from $S_{(u,v)}$; $c_{(u,v)} = c_{(u,v)} - \frac{1}{T_k}$;

      for ***each*** *link* $(u, v) \in \phi_{temp}$ **do**

        if $(u, v) \notin \phi_k$ **then**

          Insert F_k into $S_{(u,v)}$; $c_{(u,v)} = c_{(u,v)} + \frac{1}{T_k}$;

      $\phi_k = \phi_{temp}$;

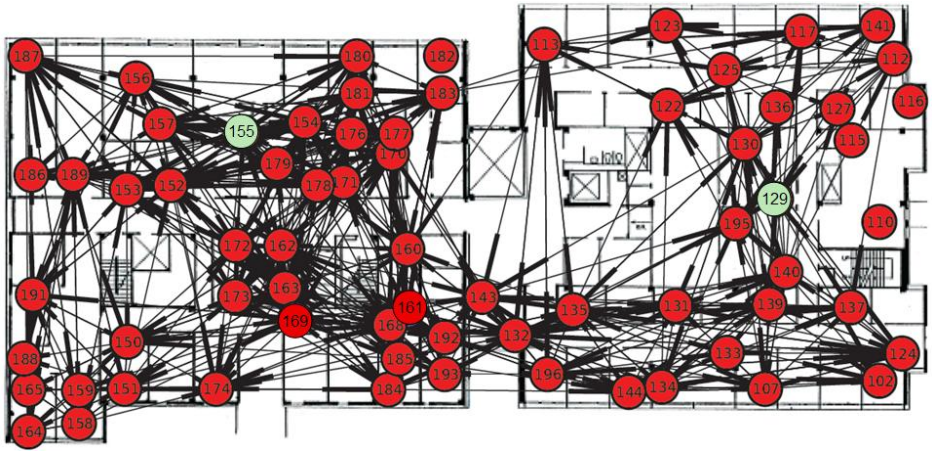
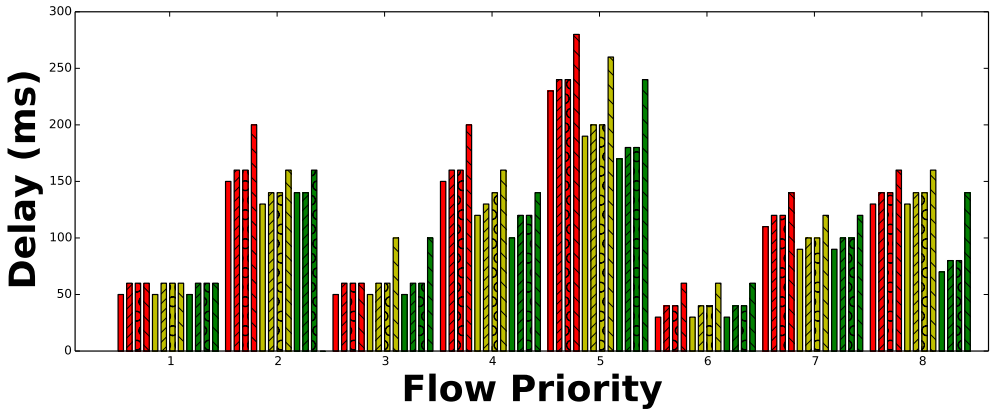


Figure 4.3: Topology of the WSAN Testbed

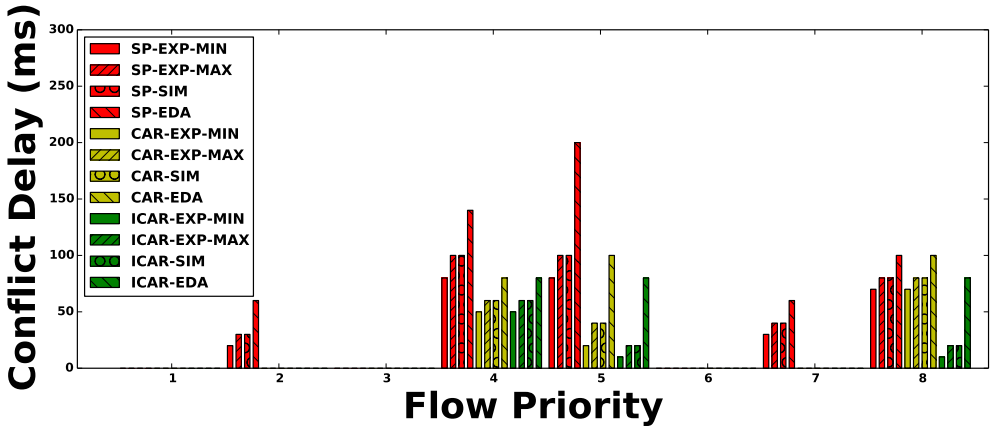
link weights for F_k. ICAR lets higher priority flows be aware of the routes of lower priority flows, and therefore reduces the overlapping of their routes. This leaves a bigger space for low priority flows and gives them a higher chance to find routes which are schedulable. If a new route R_{temp} is found, the algorithm will first check whether flow F_k with this new route is schedulable under EDA. If yes, this new route R_{temp} is assigned to F_k and flow F_k is indicated as schedulable. If not, flow F_k will use its old route R_k. If flow F_k is schedulable under EDA, we indicate it as schedulable; otherwise, F_k is unschedulable. The algorithm will enter into a new round if at least one flow is not schedulable and at least one flow has an updated route.

4.6 Evaluation

We evaluate our real-time routing algorithms through both experiments on a physical WSAN testbed and simulations based on the WSAN testbed topology. We compare our Conflict-Aware Routing (CAR) algorithm and the Iterative Conflict-Aware Routing (ICAR) algorithm with the Shortest Path Routing (SP) algorithm. In SP, each flow uses breath-first search algorithm [65] to select a route with the minimum hop count.



(a) End-to-end delay



(b) Conflict delay

Figure 4.4: Delays

4.6.1 Experiments on a WSAN Testbed

We evaluate our routing designs on an indoor WSAN testbed consisting of 63 TelosB motes, located on the fifth floors of two adjacent buildings. Figure 4.3 shows the topology of the WSAN testbed. We use motes 129 and 155 (green circles) as access points, which are physically connected to a root server (the gateway). The other motes are used as field devices (red circles). The network manager as a software runs on this root server. For each link in the testbed, we measure its *packet reception ratio (PRR)* by counting the number of received packets among 250 packets transmitted on the link. Following the practice of industrial deployment, we only add links with PRRs higher than 90% to the topology of the testbed. We implement a multi-channel TDMA MAC protocol on top of the IEEE 802.15.4 physical layer. Clocks of network devices across the entire network are synchronized using the Flooding Time Synchronization Protocol (FTSP) [143]. Time is divided into 10 ms slots.

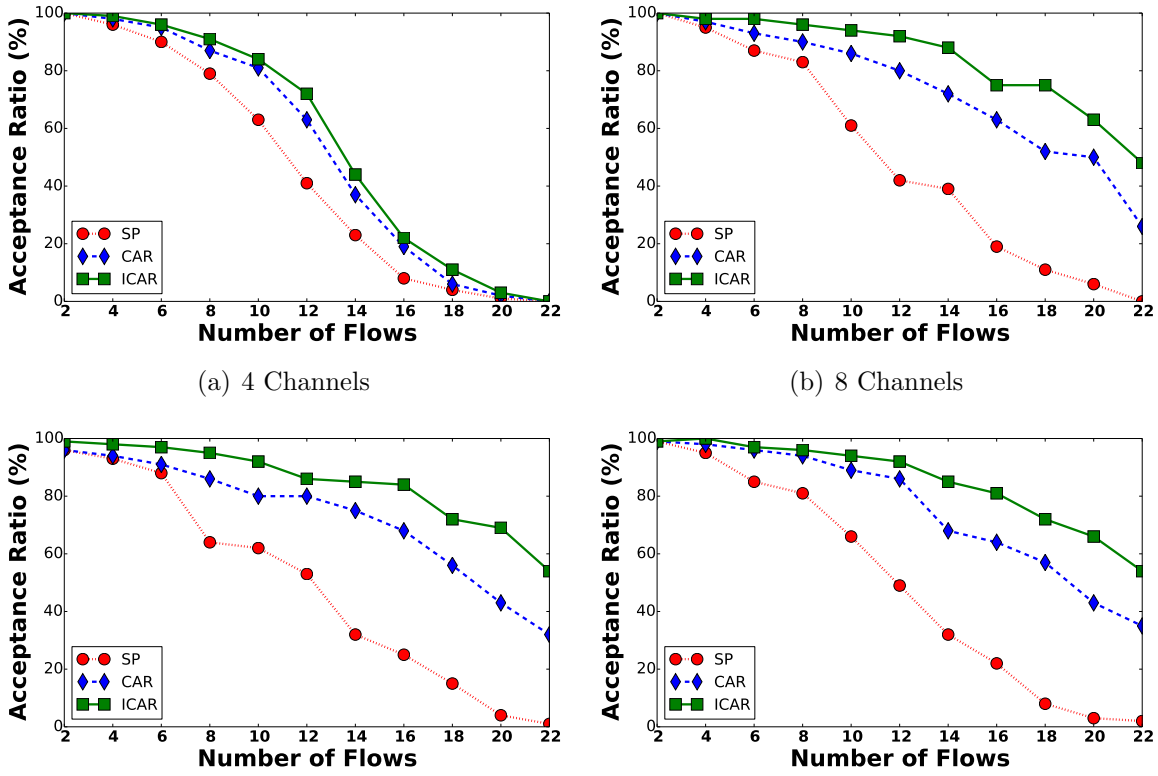
We generate 8 flows in our experiment. We use 8 channels in this experiment. The period of each flow is picked up from the range of $2^{4\sim7} \times 10$ milliseconds. The length of the hyper-period is 128 milliseconds. The relative deadline of each flow equals to its period. All flows are schedulable based on our delay analyses. We run our experiments long enough such that each flow can deliver at least 100 packets.

In Figure 4.4, we compare delays from the experimental results with delay analyses as well as simulation. We compare four delays for each flow: minimum delay in experiments (EXP-MIN), maximum delay in experiments (EXP-MAX), maximum delay in simulation (SIM), and the estimated delay in EDA [170]. We evaluate both the end-to-end delays and the conflict delays. To save space, Figure 4.4(a) shares the same legend with Figure 4.4(b).

First of all, the results show for both the end-to-end delay and conflict delay, every flow has the four delays follow the following order: EXP-MIN $\leq$ EXP-MAX $\leq$ SIM $\leq$ EDA.

This shows that simulation and delay analysis are safe upper bounds of the actual delays. In addition, SIM is consistently higher than EXP-MAX, which indicates our simulations can generate test cases with worse delays than those observed on the testbed.

Figure 4.4(a) compares end-to-end delays of flows based on different routing algorithms: SP, CAR, and ICAR. The results show CAR and ICAR can reduce the end-to-end delays



(a) 4 Channels

(b) 8 Channels

(c) 12 Channels

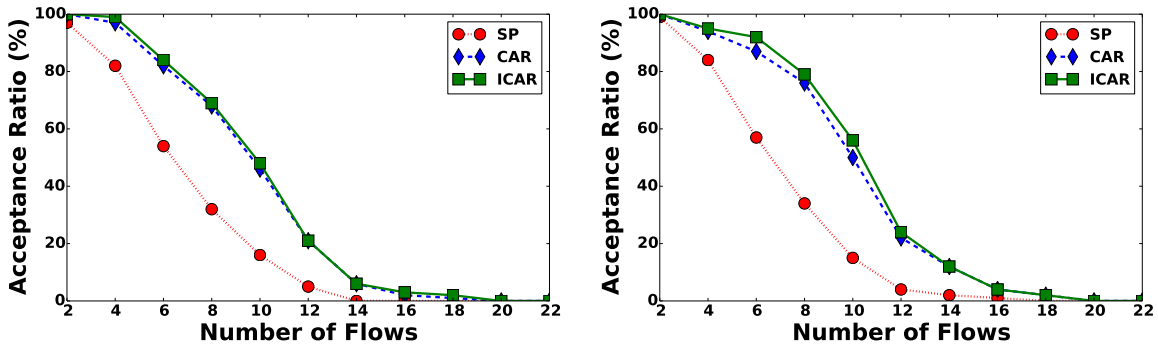
(d) 16 Channels

Figure 4.5: Acceptance Ratio in Simulation

compared with SP. Furthermore, ICAR can further reduce the delays for flows with low priorities. Given we have enough channels in this experiment, there is no contention delay. We further compare conflict delays of flows in Figure 4.4(b). Clearly, CAR and ICAR can reduce the conflict delays of flows. For example, flow 7 has conflict delays in SP routing. However, its conflict delays in CAR and ICAR routings are zero. By reducing conflict delays, CAR and ICAR can reduce the end-to-end delays of flows.

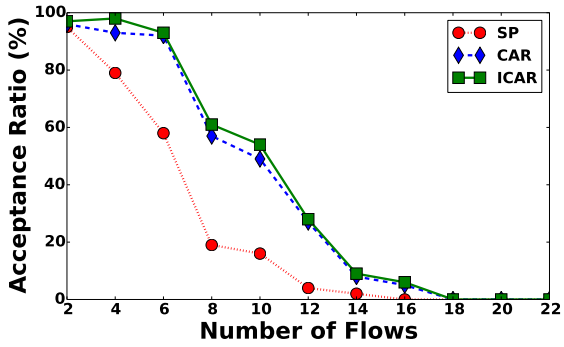
4.6.2 Simulations

Besides the testbed experiments, we also test our routing algorithms through simulations on testbed topology. The simulator uses the same routing and scheduling design used on our testbed experiments and is written in C++. All simulations are performed on a MacBook

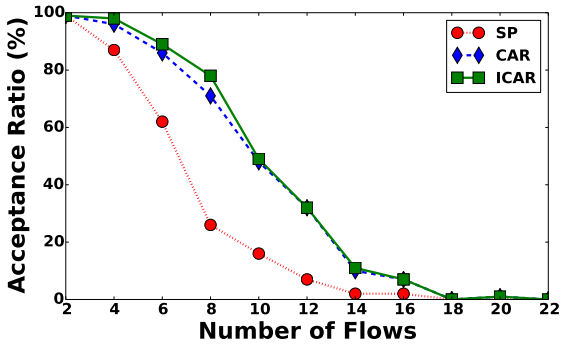


(a) 4 Channels

(b) 8 Channels

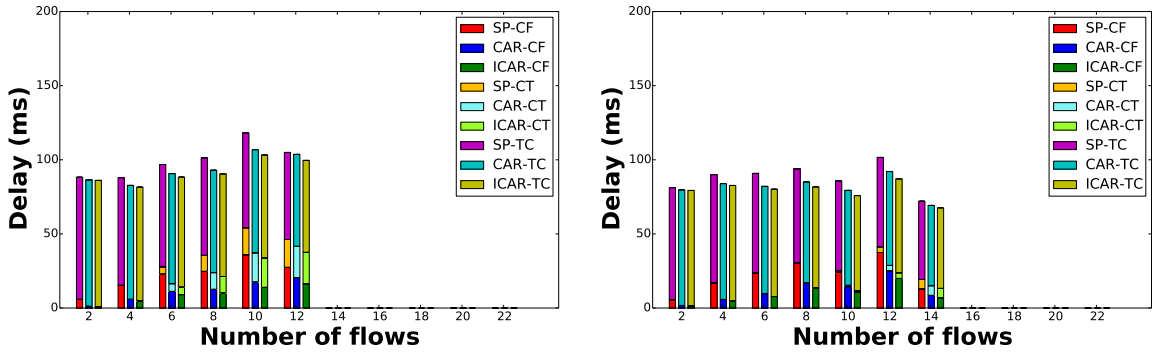


(c) 12 Channels



(d) 16 Channels

Figure 4.6: Acceptance Ratio in Analysis



(a) 4 Channels

(b) 8 Channels

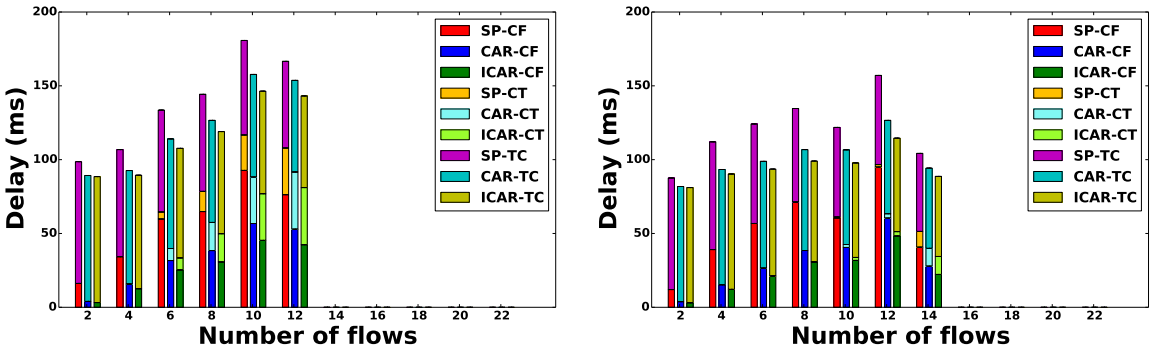
(c) 12 Channels

(d) 16 Channels

Figure 4.7: Delays in Simulation

Pro laptop with 2.4 GHz Intel Core 2 Duo processor. To show the impact of the number of channels, we test our algorithms under different number of channels (4, 8, 12, and 16) in our simulation. We test our routing designs on different numbers of flows by increasing the numbers of source and destination pairs from 2 to 22. The period T_k of the each flow F_k is randomly generated in the range of $2^{4\sim7} \times 10$ milliseconds. The relative deadline D_k of every flow F_k is equal to its period. For each flow set, we generate 100 test cases and simulate them on testbed topologies.

We first compare the acceptance ratios of CAR, ICAR and SP in Figure 4.5. SP always has the lowest acceptance ratio. Both CAR and ICAR have much higher acceptance ratios than SP when the network has at least 8 channels. ICAR has a higher acceptance ratio than CAR, which shows the benefit of letting flows with higher priorities be aware of the routes of lower priority flows. The performance of our real-time routing algorithms improves



(a) 4 Channels

(b) 8 Channels

(c) 12 Channels

(d) 16 Channels

Figure 4.8: Delays in Analysis

when the number of channels increases. Because when the network has very few channels, contention delay is the main part of end-to-end delay. However, when the network has more channels, the conflict delay becomes the dominant part of the end-to-end delay. Compared to SP, CAR and ICAR can improve the acceptance ratio by 239% and 350% in average with 16 channels, respectively. We further compare the acceptance ratios of CAR, ICAR and SP on efficient delay analysis [170] in Figure 4.6. Our simulation results show CAR and ICAR have higher acceptance ratio in delay analysis compared to SP. Because the delay analysis is pessimistic compared to simulation, acceptance ratios in delay analyses (Figure 4.6) are lower than simulation (Figure 4.5).

We further compare end-to-end delays of CAR, ICAR, and SP in Figure 4.7. Here we draw the average delays of all 100 test cases. We use CF to stand for conflict delay, CT for contention delay, and TC for transmission count (number of transmissions scheduled on the route). When the number of channels is small (4 or 8), the contention delays can be important part of the end-to-end delays. However, when the network has 12 channels, the contention delays are zero, and conflict delays dominate since then. Although CAR and ICAR may lead to routes with longer hop count, their end-to-end delays are smaller than SP in average. Because CAR and ICAR have fewer conflict delays than SP in all cases. The end-to-end delays in delay analysis [170] show the same trend in Figure 4.8.

We compare the execution time of SP, CAR, and ICAR when there are 10 channels in Figure 4.9. The execution time increases as the number of flows increases in all three algorithms. The execution time of three routing algorithms follows this order: SP<CAR<ICAR. SP has the lowest execution time since it uses the breadth-first search algorithm. ICAR has a higher execution time than CAR because it is an iterative algorithm. The execution time of ICAR is less than $200ms$ when the number of flows is 22, which is acceptable in real-world operations. We also show the number of iterations in Figure 4.10. The number of iterations increases as the number of flows increases. Even for 22 flows, the maximum number of iterations is 4 in our simulations, which is relatively small when considering the size of the network.

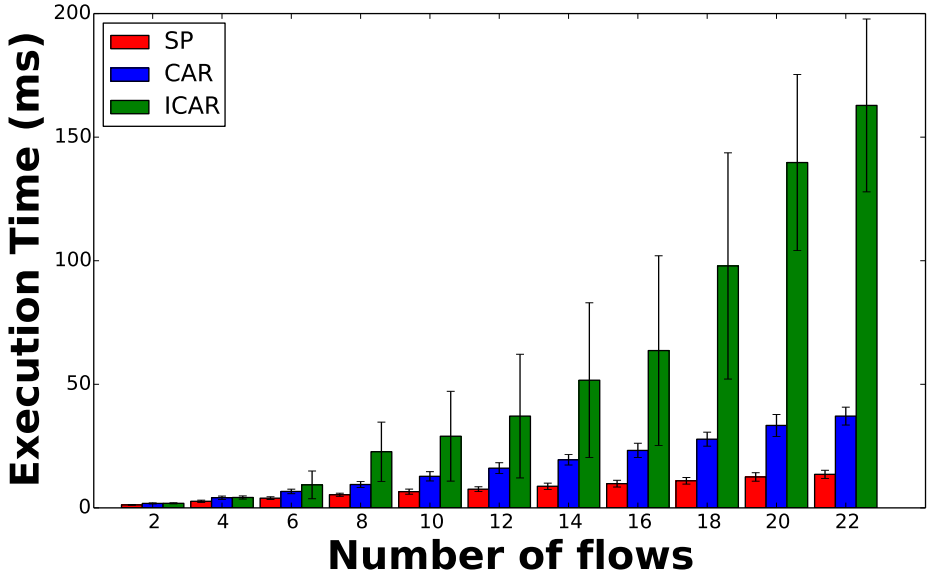


Figure 4.9: Execution Time

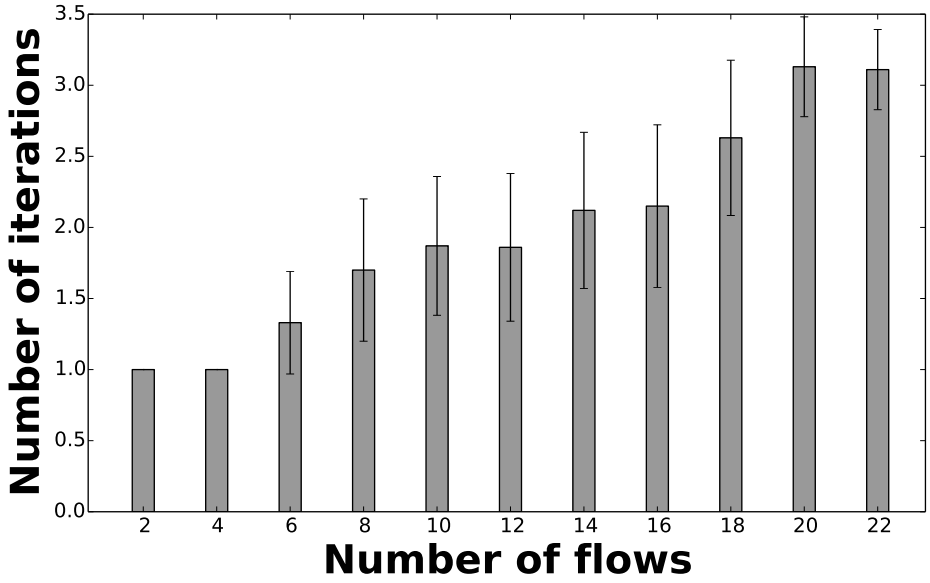


Figure 4.10: Number of Iterations

4.7 Summary

As process industries start to adopt wireless sensor-actuator networks (WSANs) for control applications, it is crucial to achieve real-time communication in this emerging class of networks. Routing has significant impacts on end-to-end communication delays in WSANs. However, despite considerable research on real-time transmission scheduling and delay analysis for such networks, real-time routing remains an open question for WSANs. This paper presents a *conflict-ware real-time routing* approach for WSANs. This approach leverage a key observation that conflicts among transmissions sharing a common field device contribute significantly to communication delays in industrial WSANs such as WirelessHART networks. By incorporating conflict delays in the routing decisions, conflict-aware real-time routing algorithms allow a WSAN to accommodate more real-time flows while meeting their deadlines. Evaluation based on simulations and experiments on a real WSANs testbed show conflict-aware real-time routing can lead to up to three-fold improvement in real-time capacity of WSANs.

Chapter 5

Energy-Efficient Routing for Wireless Sensor-Actuator Networks

Process industries are adopting wireless sensor-actuator networks (WSANs) as the communication infrastructure. The dynamics of industrial environments and stringent reliability requirements necessitate high degrees of fault tolerance in routing. WirelessHART is an open industrial standard for WSANs that have seen world-wide deployments. WirelessHART employs graph routing schemes to achieve network reliability through multiple paths. Since many industrial devices operate on batteries in harsh environments where changing batteries are prohibitively labor-intensive, WSANs need to achieve long network lifetime. To meet industrial demand for long-term reliable communication, this chapter studies the problem of maximizing network lifetime for WSANs under graph routing. We formulate the network lifetime maximization problem for WirelessHART networks under graph routing. Then, we propose the optimal algorithm and two more efficient algorithms to prolong the network lifetime of WSANs. Experiments in a physical testbed and simulations show our linear programming relaxation and greedy heuristics can improve the network lifetime by up to 50% while preserving the reliability benefits of graph routing.

5.1 Introduction

With the emergence of industrial standards such as WirelessHART [26] and ISA100 [16], process industries are adopting *Wireless Sensor-Actuator Networks (WSANs)* that enable sensors and actuators to communicate through low-power multi-hop wireless mesh networks

[184]. The process industry has installed more than 1800 WirelessHART networks and 10 thousands WirelessHART devices around the world, with more than 3 billion operating hours in the field [13].

The limited energy supply of the network devices necessitates the efficient utilization of battery power. Energy consumption is closely coupled with route selection. Selecting a routing path that optimizes energy efficiency can lead to a longer network lifetime. In industrial environments, changing batteries can be dramatically expensive and difficult, e.g., oil fields spanning large areas under harsh environmental conditions. Thus, maximizing the lifetime of the network is an important problem that needs to be tackled.

To support reliable communication over wireless mesh networks, the WirelessHART standard adopts a graph routing approach. A graph route consists of a primary path and multiple backup paths. For each link on the primary path, a backup path is used to handle the link failure. Graph routing introduces unique challenges in energy-efficient routing that has not been investigated in earlier research on energy-efficient routing for wireless sensor networks.

This chapter addresses the network lifetime maximization problem under graph routing. We first formulate the lifetime maximization problem under graph routing based on the WirelessHART standard. We then propose approximation solutions based on linear programming relaxation and greedy heuristics. Specifically, our contributions are five-fold:

- Formulation of the network lifetime maximization problem under graph routing and proof of its NP-completeness.
- An optimal network lifetime maximization algorithm based on integer programming.
- An approximation algorithm through linear relaxation of the integer programming algorithm.
- An efficient greedy heuristics with lower computational complexity.
- Implementation and evaluation of the proposed algorithms on a wireless sensor-actuator network testbed, as well as in large-scale simulations.

Our evaluation shows that the linear programming relaxation and greedy heuristics can greatly improve the network lifetime by up to 50%, and that the greedy heuristic is more efficient than the linear programming relaxation approach.

The rest of the chapter is organized as follows. Section 5.2 reviews the related works. Section 5.3 describes the energy consumption model. Section 5.4 formulates the lifetime maximization problem and proves it is NP-complete. Section 5.5 presents our lifetime maximization graph routing algorithms. Section 5.6 evaluates different graph routing algorithms in experiments and simulations. Section 5.7 concludes the chapter.

5.2 Related Work

There has been increasing interest in developing new routing approaches for WSANs. For example, Han et al. [94] proposed routing algorithms to build graph routes, but this work does not address energy efficiency of graph routing. Energy-aware routing for wireless sensor and *ad hoc* networks has received significant attention [178]. Stojmenovic and Lin [189] proposed a protocol to minimize total power consumption and extend network lifetime. Chang and Tassiulas proposed to maximize the network lifetime by balancing network traffic among the nodes in proportion to their residual energy [50–52]. Li et al. [135] proposed a routing protocol that combines the benefits of selecting the path with minimum power consumption and the path that maximizes residual power in the nodes. Doshi et at. [68] implemented a minimum energy routing version of the DSR protocol in a network simulator. Kalpakis et al. [108] studied lifetime maximization problem for tree topology network. In addition, energy-aware geographic routing was studied in [126, 139–141, 144, 174, 204, 213].

Despite considerable results on the general problem of network lifetime optimization, none of the aforementioned work addresses graph routing, which is an important technique that WirelessHART standard used to achieve reliable communication in industrial settings [176]. Hence they are not applicable for WSANs. To meet this open challenge in industrial WSANs, we investigate the problem of network lifetime maximization under graph routing in WirelessHART networks in this chapter.

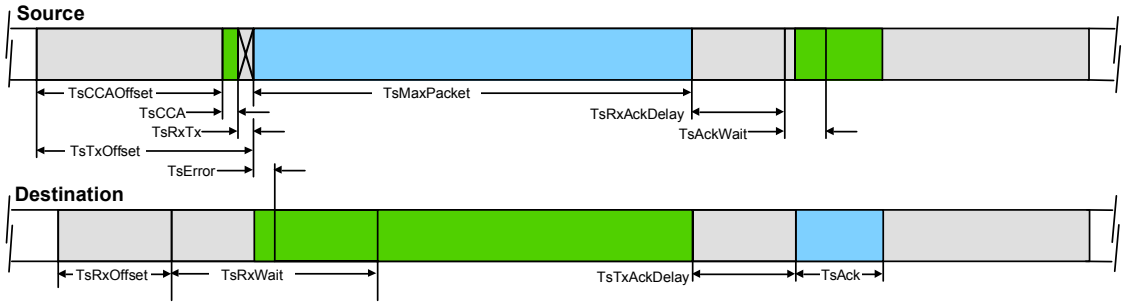


Figure 5.1: Transaction timing in one time slot [26]

5.3 Energy Consumption Model

We analyze the energy consumption under graph routing in this section. Specifically, for a single packet, we analyze the energy consumption of each device on both the primary path and backup paths. Since the scheduling policies for transmission on the primary path and backup path are different, we analyze the energy consumption for them separately. For each transmission on the primary path, two dedicated slots are assigned. If the first transmission succeeds, the following retransmission will not occur and both sender and receiver will turn off their radio at the second time slot. Otherwise, a second retry will happen in the second assigned time slot. If both of the transmissions on the primary path fail, there will be a third retry on the backup path.

Figure 5.1 shows the timing of a transmission in a time slot according to the WirelessHART standard [26]. The top of the timing diagram shows the operation of the sender and the bottom shows that of the receiver. When a shared slot is assigned, the sender will perform the Clear Channel Assessment (CCA) before transmitting the packet. We use *TsMaxPacket* to denote the amount of time it takes to transmit the longest possible message. When scheduled as the transmission's receiver, the receiver must enter receive mode. The receiver must keeping radio on to listen potential packet transmission. We denote the minimum time to wait for start of message as *TsRxWait*. If an intended communication is detected, the receiver keeps listening until it has received the whole packet. Otherwise, the receiver will turn off the radio after the receive window expires. We denote the power of transmitting and receiving a packet as P_t and P_r respectively.

Assume v_i is a network device on the primary path which is scheduled to send one packet to a network device v_j. We use α to denote the Packet Reception Ratio (PRR) for this link. Then the probability that it can successfully transmit a packet to its receiver v_j on the first try is α. The probability that it fails in the first try and needs to transmit the packet to v_j in the second retry is $1-\alpha$. So the expected time length that the sender keeps its radio on is

$$\alpha \times TsMaxPacket + 2(1-\alpha)\, TsMaxPacket = (2-\alpha)\, TsMaxPacket.$$

A receiver on the primary path has the same expected time length keeping its radio on for the same reason. By incorporating the current, we get the expected energy consumption of a network device as a sender or a receiver for delivering one packet on the primary path. We denote E_t as the expected energy consumption of device v_i to transmit a packet to v_j on a primary path, thus

$$E_t = (2-\alpha)P_t \times TsMaxPacket. \tag{5.1}$$

The expected energy consumption of device j to receive a packet from i on a primary path is:

$$E_r = (2-\alpha)P_r \times TsMaxPacket. \tag{5.2}$$

Since transmission on a backup path will happen only when the two consecutive dedicated transmissions fail, the chance that there is an actual packet transmission on a backup path is $(1-\alpha)^2$ (e.g. less than 0.01 if we use a PRR threshold of 0.9). However, as long as a transmission is scheduled on a link, the receiver needs to turn on the radio and listen for $TsRxWait$ time to check whether there is an incoming packet. Then the expected energy consumption of device i on a backup path to transmit a packet is

$$E_{tb} = (1-\alpha)^2 P_t \times TsMaxPacket. \tag{5.3}$$

The expected energy consumption of device i to receive a packet on backup path is:

$$E_{rb} = (1-\alpha)^2 P_r \times TsMaxPacket + (1-(1-\alpha)^2)P_r \times TsRxWait. \tag{5.4}$$

Table 5.1 summarizes the power to transmit or receive a packet on the CC2420 radio chip [11], which is compatible to IEEE 802.15.4 standard (physical layer of WirelessHART standard). Table 5.1 also shows the timing parameters in packet transmission defined in the WirelessHART standard [26]. Based on Table 5.1, we obtain the expected energy consumptions in Table 5.2, assuming a PRR of 90%, a typical threshold used for blacklisting links in industrial WSANs.

Parameter	Value	Unit
P_t	52.2	mW
P_r	59.1	mW
$TsMaxPacket$	4256	μs
$TsRxWait$	2200	μs

Table 5.1: Representative Radio Parameters

Variable	Value	Unit
E_t	277	μJ
E_r	244	μJ
E_{tb}	2.2	μJ
E_{rb}	131	μJ

Table 5.2: Expected energy consumption of devices to transmit or receive a packet

Note the expected energy consumption to transmit a packet on a backup link is less than 1% of the expected energy for transmitting a packet on a primary link. We will ignore E_{tb} in the remaining part of this chapter.

5.4 Graph Route Lifetime Maximization Problem

In this section, we formulate the *Graph Route Lifetime Maximization* (GRLM) problem. Our objective is to maximize the lifetime of the network.

Definition 2. *The network lifetime is the time it takes the first field device to exhaust its battery.*

In terms of lifetime optimization, the most significant difference between WSANs and traditional wireless sensor networks is the route diversity. Instead of scheduling transmissions on only one path, WSANs schedule transmissions on both the primary path and backup paths.

Definition 3. *In a GRLM problem, we are given a graph* $G = (V, E)$ *with* battery B_{v_i} *for each device* v_i*, and a set of flows* $F = \{f_1, f_2, \cdots, f_N\}$*. Each flow* f_k *has a source* s_k*, a destination* d_k*, and a rate* r_k*. The GRLM problem is to find graph routes for all flows to maximize the network lifetime.*

The GRLM problem is NP-complete because even the source routing version of the problem is NP-complete as shown below.

Proof. To prove the SRLM problem is NP-complete, we prove its decision version is NP-complete. The decision problem of SRLM is given a network lifetime T, whether this network lifetime T can be satisfied.

We begin with the work of Fortune et al. [76], which proved the *Maximum Edge-Disjoint Paths* problem (MEDP) is NP-complete. In MEDP, we are given an graph $G = (V, E)$, and a set of k device pairs $\Gamma = \{(s_i, t_i) : i = 1, \cdots, k\}$. The goal is to find the maximum subset of pairs from Γ , along with a path for each chosen pair, so that no two paths share the same link. The decision problem of MEDP is whether a given set of device pairs in Γ can satisfy the requirements.

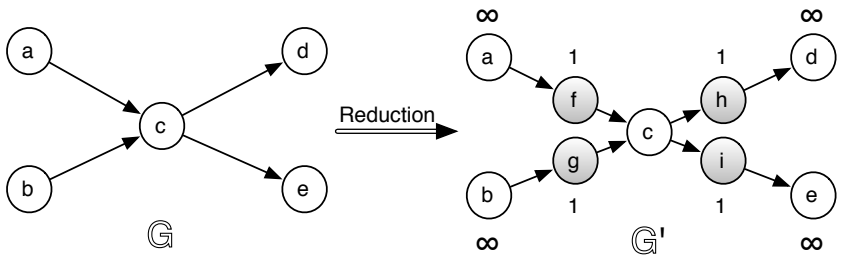


Figure 5.2: Reduction

Clearly, the decision problem of SRLM is in NP since we can verify in polynomial time if a candidate solution provides source routes for all flows and achieves the targeted lifetime.

To show the problem is NP-complete, we reduce it from MEDP, which is known to be NP-complete [76]. The reduction algorithm takes an instance of the MEDP problem as input. Given a graph G, we construct an auxiliary graph G' in the following manner. For each link e in G (i.e., $a \to c$ in Figure 5.2), we break it into two links ($a \to f$ and $f \to c$) and add a new link-device (f) to connect these two links (Figure 2). All devices in the original graph are assigned with battery capacity as $+\infty$, and all newly added devices are assigned with unit battery capacity 1. For each device pair (s_i, t_i) in Γ, we create a flow F_i in G' with source s_i, destination t_i, and unit rate 1. The targeted lifetime of the network is $T = \frac{1}{E_t+E_r}$. Note that $\frac{1}{E_t+E_r}$ is the lifetime of a link-device if only one flow goes through it. To complete the proof, we show that all pairs in Γ have link-disjoint paths if and only if the network lifetime of G' is no less than T.

$\leftarrow$ If all device pairs have link-disjoint paths in G, then the reduced paths in G' can have a network lifetime no less than T. Since at most one reduced flow goes though each link-device and the lifetime of each link-device is no smaller than the network lifetime target T, the network lifetime of G' is no smaller than T.

$\rightarrow$ If the network lifetime of G' is no less than T, then there are link-disjoint paths for all device pairs in Γ. Since the battery of each link-device can support exactly one flow, only one path will go through each link-device, which indicates those paths are edge-disjoint paths. Then we get link-disjoint paths in the original graph G.

Given the reduction is in polynomial time and a instance of MEDP is true if and only if the reduced instance of SRML is true, we prove that SRML is NP-hard, as well as NP-complete. □

5.5 Lifetime Maximization Graph Routing Algorithms

In this section, we propose an optimal solution based on integer programming, followed by more efficient solutions based on linear relaxation and a greedy heuristics.

5.5.1 Integer Programming

In this subsection, we formulate the GRLM problem into an integer programming based on our energy consumption model.

All the field devices are powered by batteries, while access points and the Gateway are tethered to wired power sources. The lifetime of a field device is modeled as the initial battery divided by its average power consumption, also refereed as load in this chapter. Here we denote the initial battery capacity of a device v_i as B_i, and the load as L_i. For access points and the Gateway, batteries are set to be infinity. Our goal here is to maximize the minimum lifetime among all devices, which is expressed as $\max \min_i \frac{B_i}{L_i}$. This objective function can be transformed to minimize the maximum normalized load γ_i, defined as $\frac{L_i}{B_i}$ for device v_i. Hence the GRLM problem can be formulated as $\min \max_i \gamma_i$.

We formulate the integer programming as follows. The primary path variable $x_{i,j}^k$ is a binary variable. If link $\overrightarrow{ij}$ is used in the primary path for flow k, then $x_{i,j}^k$ equals 1, otherwise, it equals 0. The same rule is applied to backup path variable $y_{i,j}^k$. However, since multiple backup paths may share a same link, the backup path variable $y_{i,j}^k$ is an integer variable, which could be larger than 1.

First, there is only one link used in the primary path among all outgoing links of the source s_k (5.5a). Then the conservation constraint (5.5b) says the sum of outgoing primary path variables equals the sum of incoming primary path variables at every device except the source and the destination, where $\delta_{i,j}$ is the Kronecker delta function [17]. Here $\delta_{i,j}$ equals 1 if i and j are the same, and 0 otherwise.

Objective: minimize Γ

$$\sum_{\overrightarrow{s_k j} \in E} x^k_{s_k,j} = 1 \tag{5.5a}$$

$$\sum_{\overrightarrow{ji} \in E} x^k_{j,i} + \delta_{i,s_k} = \sum_{\overrightarrow{ij} \in E} x^k_{i,j} + \delta_{i,t_k}, \forall i \in V \tag{5.5b}$$

$$\sum_{\overrightarrow{ji} \in E} y^k_{j,i} + \sum_{\overrightarrow{ij} \in E} x^k_{i,j} = \sum_{\overrightarrow{ij} \in E} y^k_{i,j}, \forall i \in V \setminus \{t_k\} \tag{5.5c}$$

$$\sum_{\overrightarrow{ip} \in E: p != j} y^k_{i,p} >= x^k_{i,j}, \quad \forall \overrightarrow{ij} \in E \tag{5.5d}$$

$$\gamma_i = \sum_k \frac{r_k}{B_i} (\sum_{\overrightarrow{ji} \in E} x^k_{i,j} E_t + \sum_{\overrightarrow{ij} \in E} x^k_{j,i} E_r + \sum_{\overrightarrow{ji} \in E} y^k_{j,i} E_{rb}) \tag{5.5e}$$

$$\gamma_i \leq \Gamma, \forall i \in V \tag{5.5f}$$

$$x^k_{i,j} \in \{0,1\}, y^k_{i,j} \in \mathbb{Z}, \forall \overrightarrow{ij} \in E \tag{5.5g}$$

The conservation constraint for backup path variables is different from the constraint for primary path variables, because backup paths do not start from the source of the flow, instead, they start from devices on the primary path. For backup paths, there are two cases. For a device on a backup path but not on the primary path (e.g. network device z in Figure 2.2(b)), it follows the same conservation constraint as the primary path variables, which means the sum of outgoing backup path variables equals to the sum of incoming backup path variables. For a network device which is on both the backup path and primary path (e.g. u in Figure 2.2(b)), it does not have any incoming backup path. However, it still has an outgoing backup path, and the amount of backup path variables equals to the amount of outgoing primary path variables. To incorporate both cases, we formulate this requirement in constraint (5.5c). Basically, the sum of outgoing backup path variables from a device equals to the sum of incoming backup path variables plus the sum of outgoing primary path variables.

Also, since backup link should not coincide with the primary link for the same packet, constraint (5.5d) is added to make sure that the backup path of a link on the primary path does not use this link. Constraint (5.5e) calculates the normalized load γ_i of each device i.

And constraint (5.5f) guarantees that normalized loads of all network devices are no larger than Γ. The objective is to minimize the maximum normalized load γ, which is same to minimize Γ.

5.5.2 Linear Programming Relaxation

Given the problem is NP-complete, the integer programming approach is extremely slow. We use a linear programming relaxation approach to propose a faster approach. We solve the problem in two steps. In the first step we focus on the primary path variables. In the beginning, we relax each primary path variables $x_{i,j}^k$ from binary to real number within $[0, 1]$, and relax each backup path variable $y_{i,j}^k$ from integer to non-negative real number. Then we solve the problem and obtain the solution. We round the variables to 1 if it's above a threshold θ, otherwise round it to 0. We want to find the highest threshold θ for primary path variables such that there exists a path from the source to the destination. We use a binary search algorithm to find this threshold. The initial threshold is 0.5. If a path is found, then we increase the threshold to 0.75. Otherwise, we decrease the threshold to 0.25. The binary search algorithm terminates if at least one path is found under a threshold and the step size is less than 0.05.

After the first step, we obtain primary paths for all flows. In the second step, we keep primary path variables fixed and relax backup path variables to non-negative real numbers. Following the same approach in the first step, we use a binary search algorithm to find the highest threshold that we can find a path from the source to the destination. We use the GNU Linear Programming Kit (GLPK) [14] to solve the integer programming and its linear programming relaxation.

5.5.3 Greedy Heuristic

Although we use linear relaxation to speed up the integer programming approach, in practice it is still not efficient enough for large networks. In this subsection, we introduce a greedy heuristic which finds graph routes that lead to high network lifetime and remain computationally efficient. By incorporating the traffic load and battery capacity into the greedy

algorithm, the greedy heuristic picks up a graph route with small normalized load. Our greedy heuristic runs iteratively. In each iteration, we pick up graph routes for flows from the highest rate to the lowest rate. For each flow, we pick up a graph route with minimum load. Our iterative algorithm stops if no improvement can be found in an iteration.

The function the greedy heuristic calls in each iteration is named as Minimum Load Graph Route (MLGR) and is presented in Algorithm 5. We use an algorithm like Dijkstra's shortest path algorithm. We use λ to record temporary normalized loads for all devices in the network. We maintain a queue Q which includes all network devices that have an updated normalized load. We also maintain a set Φ, which includes all devices that can be added into the primary path.

At each step, a device u with minimum normalized load λ_u is picked up from the queue. If its normalized load λ_u equals ∞, then the remaining devices cannot be added to the primary path. Then MLGR function fails to find a graph route for current flow and returns ∞. If u is the source, then the MLGR function adds it to the primary path and returns its normalized load λ_u. We can obtain the primary path by tracing back through H, and obtain the backup paths with P.

If none of above case is true, we will check u's neighbors one by one to see whether they can be added into the primary path. For each neighbor v, we use the Minimum Load Source Route (MLSR) function in Algorithm 6 to check whether there is a path from v to the destination d in the graph $G' = (V, E \setminus \{\overrightarrow{vu}\})$ and return the one with the minimum normalized load. We update the normalized load of device v based on its new normalized load $\gamma_v + \frac{rE_t + rE_r}{B_v}$, its parent u's normalized load λ_u and the normalized load of the backup path.

Here the MLSR function is a single path version of MLGR. At each step, it picks up the device u with minimum normalized load λ_u. If λ_u equals ∞, then the source s cannot be connected to the destination d, and MLSR function returns ∞. If the source s is picked up with a normalized load λ_s less than ∞, then s is connected with the destination d, and MLSR function returns λ_s. The MLSR function can obtain the path from the last hop vector H. If none of above case is true, the MLSR function will check device u's neighbors and update their normalized load according to u's normalized load λ_u.

Algorithm 5: Minimum Load Graph Route

Function *MLGR*(G, s, d, r, γ, B)

Input : A graph $G(V, E)$, source s, destination d, flow rate r, normalized load vector γ, battery vector B

Variable: Last hop vector H, Backup Paths P, temporary normalized load λ

Output : Normalized load of the graph route picked up by the algorithm (∞ if no graph route is found)

```
for each vertex v ∈ V do
    λ_v = ∞;
    H_v = NULL;
    P_v = ∅;
    add v to Q;
λ_d = γ_d + rE_r / B_d;
while Q is not empty do
    u = v ∈ Q with minimum λ_v;
    remove u from Q;
    if λ_u == ∞ then
        return ∞;
    if u == source then
        return λ_u;
    for each neighbor v of u within Q do
        Graph G' = (V, E \ {v⃗u});
        temp = MLSR(G', v, d, P_v, r, γ, B);
        if temp ≠ ∞ then
            alt = max(λ_u, γ_v + (rE_t + rE_r) / B_v, temp);
            if alt < λ_v then
                λ_v = alt;
                H_v = u;
```

Algorithm 6: Minimum Load Source Route

Function *MLSR*$(G, s, d, P_s, r, \gamma, B)$

Input : A graph $G(V, E)$, source s, destination d, flow rate r, normalized load vector γ, battery vector B

Variable: Last hop vector H, temporary normalized load λ

Output : Normalized load of the source route picked up by the algorithm (∞ if no graph route is found)

for ***each*** *vertex* $v \in V$ **do**
- $\lambda_v = \infty$;
- $H_v = NULL$;
- add v to Q;

$\lambda_d = \gamma_d + \frac{rE_{rb}}{B_d}$;

while Q ***is not*** *empty* **do**
- $u = v \in Q$ with minimum λ_v;
- remove u from Q;
- **if** $\lambda_u == \infty$ **then**
  - return ∞;
- **if** $u ==$ *source* **then**
  - return λ_u;
- **for** ***each*** *neighbor* v *of* u *within* Q **do**
  - $alt = max(\lambda_u, \gamma_v + \frac{rE_{rb}}{B_v})$;
  - **if** $alt < \lambda_v$ **then**
    - $\lambda_v =$ alt;
    - $H_v = u$;

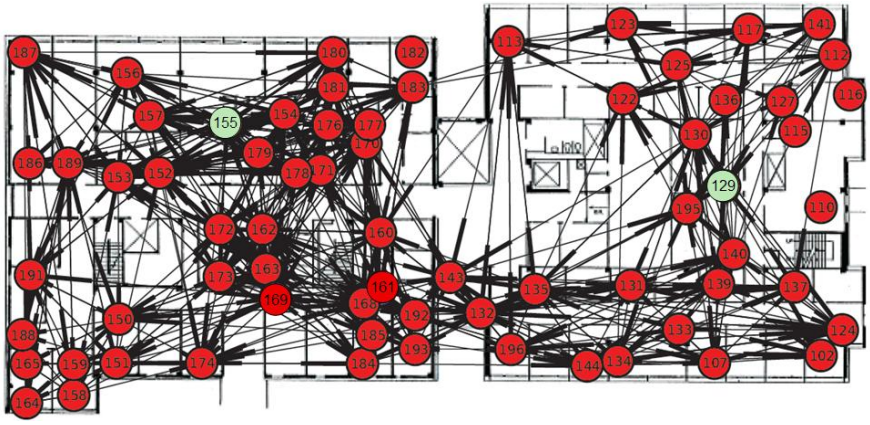


Figure 5.3: Topology of the WSAN Testbed

5.6 Evaluation

We evaluate our routing algorithms through both experiments on a physical WSAN testbed and simulations. We compare our Integer Programming approach (IP), Linear Programming approximation (LP), and Greedy Heuristic algorithm (GH) with the reliable and real-time routing (RRC) approach that Han et al. proposed in [94] and Dijkstra's shortest path algorithm (SP) [65]. RRC builds uplink and downlink routing graphs for all flows based on hop count. We build a graph route on top of RRC's routing graph by picking up on path as the primary path and use available alternative paths as backup paths. Because RRC does not fully explore the network to find backup paths, some network devices on the primary path don't have backup paths. In SP, we first run Dijkstra's algorithm to get the primary path with shortest hop count, then run the same algorithm to pick up backup paths for each network device on the primary path.

5.6.1 Experiments on a WSAN Testbed

We evaluate our routing designs on an indoor WSAN testbed consisting of 63 TelosB motes, located on the fifth floors of two adjacent buildings. Figure 5.3 shows the topology of the WSAN testbed. We use motes 129 and 155 (green circles) as access points, which are physically connected to a root server (Gateway). The other motes are used as field devices (red circles). The network manager as a software runs on this root server. The rest of motes

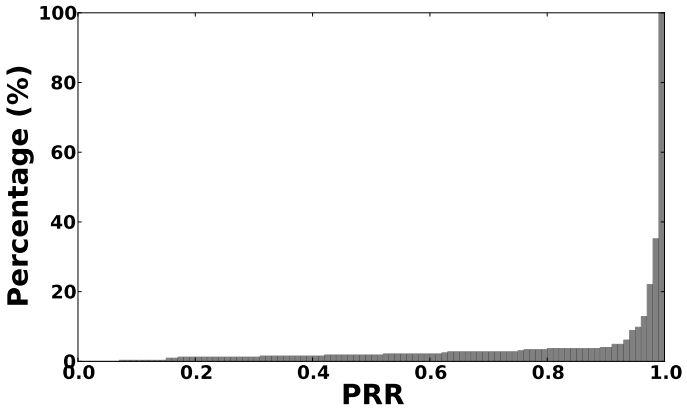


Figure 5.4: Cumulative Histogram of Link Qualities

Routing	Algorithm	Flow Index							
		1	2	3	4	5	6	7	8
Source	SP	0.993	0.874	0.898	1.0	1.0	1.0	1.0	1.0
	RRC	0.992	0.760	0.833	0.996	0.989	0.994	1.0	1.0
	GH	0.996	0.886	0.897	0.997	0.998	1.0	1.0	1.0
	LP	0.997	0.827	0.896	0.998	0.989	1.0	1.0	1.0
Graph	SP	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	RRC	0.996	0.990	0.988	1.0	1.0	1.0	1.0	1.0
	GH	0.998	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	LP	0.998	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Table 5.3: Delivery Ratios of Flows

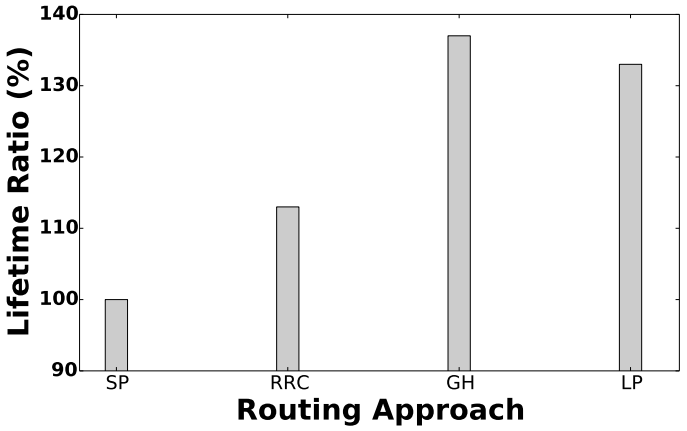


Figure 5.5: Expected Network Lifetime relative to SP

work as network devices. For each link in the testbed, we measured its *packet reception ratio (PRR)* by counting the number of received packets among 250 packets transmitted on the link. Following the practice of industrial deployment, we only add links with PRR higher than 90% to the topology of the testbed. To avoid channels occupied by the campus Wi-Fi, we use IEEE 802.15.4 channel 11 to 15 in our experiments. We implement a multi-channel TDMA MAC protocol on top of the IEEE 802.15.4 physical layer. Clocks of network devices across the entire network are synchronized using the Flooding Time Synchronization Protocol (FTSP) [143]. Time is divided into 10 ms slots.

We generate 8 flows in our experiment. The period of each flow is picked up from the range of $2^{0\sim7}$ seconds, which are typical periods used in process industry as defined in WirelessHART standard [26]. The length of the hyper-period is 128 seconds. The relative deadline of each flow equals to its period. We run the experiments for 100 rounds of hyper-period (around 3 hours) to collect at least 100 periods of data traces for each flow. Based on the data traces we collected, we evaluate our proposed approaches in terms of *delivery ratio* and *expected network lifetime*. The delivery ratio of a flow is defined as percentage of packets that are successfully delivered to destination compared to total number of packets.

The expected network lifetime is calculated based on the collected traces. Because TelosB motes in the testbed are wired powered, we assign virtual battery capacity for each mote randomly from $8000J$ to $9000J$, where $8640J$ is the typical capacity of two AA batteries. We analyze the collected data traces from the experiments to obtain the energy consumption of each network device in 100 rounds of hyper-period. Based on that, we project the expected network lifetime.

To study the reliability, we first measure the link qualities in our test. Figure 5.4 shows the cumulative histogram of link qualities (PRR) of 327 links we used in our experiments. Here we collect link qualities of each link on all 4 channels, so there are 1380 data points in total. Although our link selection process only picks up links with PRR higher than 90%, we find some links have much lower PRR than the 90% threshold at run time. For example, link $\overrightarrow{158\ 156}$ under channel 12 has the lowest PRR of 7%. The dynamics of wireless links suggest it is necessary to have route redundancy.

Table 5.3 shows the delivery ratios of all 8 flows. Here we compare the delivery ratio of both graph routes as well as the source routes. We use the primary path of the graph route as

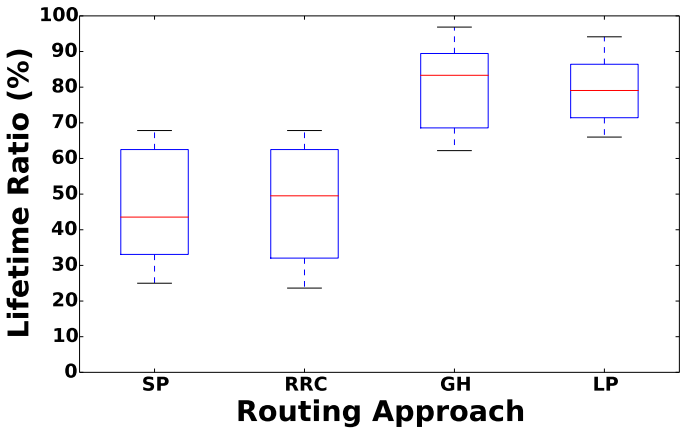


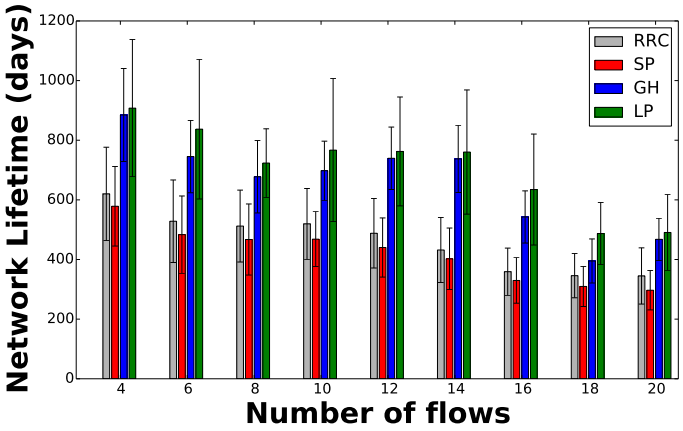
Figure 5.6: Expected Lifetime Relative to Optimal Solution

the source route of each flow. We first compare graph routing with source routing. Our results show that graph route provides better delivery ratio than source route. For example, the delivery ratios of all four routing algorithms for flow 2 under source routing are below 0.9. In comparison, their delivery ratios under graph routing are at lest 0.99. Our results demonstrate the effectiveness of redundant routes in improving reliability. We also compare the delivery ratios of different routing algorithms under graph routing. RRC has the lowest delivery ratios for flows 1, 2, and 3. That's because in RRC's graph routes for flow 1, 2, and 3, 50% of the links on the primary paths don't have backup paths. The lack of backup paths make RRC vulnerable to link dynamics.

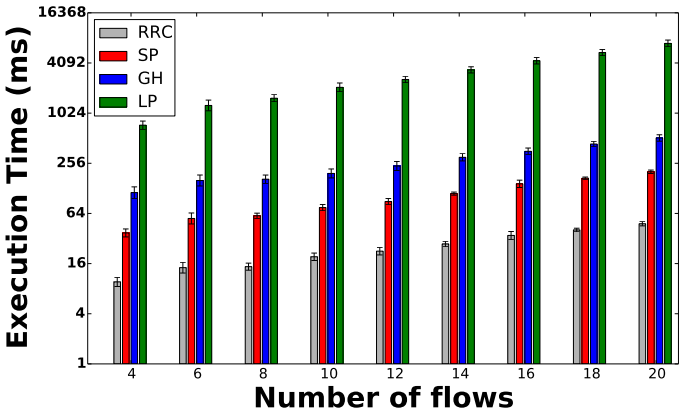
Figure 5.5 represents the expected lifetimes of four routing approaches in one experiment. The figure presents the expected lifetime ratios of different routing approaches compared to SP. The results show SP has the shortest expected lifetime and GH has the longest expected lifetime. GH's expected lifetime is 37% longer than SP, and LP's expected lifetime is 33% longer than SP. RRC has a lifetime longer than SP and shorter than LP. Our results show GH and LP are better than SP and RRC in terms of expected network lifetime.

5.6.2 Simulations

Because our Integer Programming approach (IP) is computational complex, it doesn't halt in 24 hours given the testbed topology. We evaluate all five routing algorithms in a small



(a) Network Lifetime



(b) Execution Time

Figure 5.7: Simulation Results on Testbed Topology

scale simulation. We use a subset of our testbed with 10 motes and 20 links in this small scale simulation. The simulator shares the same setup with our testbed and is written in C++. All simulations are performed on a MacBook Pro laptop with 2.4 GHz Intel Core 2 Duo processor. We use a trace driven simulation. On each link, we collect results of 250 packet transmissions. We use the first 100 packets to obtain the link quality (PRR) of the link, then use the remaining 150 packets as the evaluation trace. We use links with at least 90% PRR in the simulation. In the simulation, when a packet is transmitted on a link, the simulator will pickup a data point from the evaluation trace. We generate different results by randomly generating network device battery capacities from $8000J$ to $9000J$.

Figure 5.6 shows the lifetime ratios of SP, RRC, GH, and LP compared to IP. Our results show GH and LP always have expected network lifetime higher than 50%. The median of GH and LP are 83% and 79%. Compared with IP, SP and RRC have median lifetime ratios as 44% and 47%. The figure shows GH and LP have good expected lifetime compared the IP approach and greatly outperform SP and RRC.

Besides small scale simulations, we also test our algorithms with large number of flows in simulation on whole testbed topology. We evaluate our routing designs on different numbers of flows by increasing the number of sensor and actuator pairs. We compare four routing designs in terms of *network lifetime* and *execution time*.

We showed the network lifetime of different routing designs in Figure 5.7(a). In general, network lifetime decreases as the number of flows increases. Because more flows bring more energy consumption to network devices. Furthermore, results show SP always gives the shortest network lifetime, RRC is longer than SP but shorter than GH and LP. GH and LP provide longer network lifetime than the other two. The figure shows GH and LP can improve the network lifetime from SP by up to 43% and 51%.

The computational complexity of four routing algorithms are presented in Figure 5.7(b). The results show LP is much slower than other three algorithms. Because linear programming solver in general is slower than straightforward routing algorithms such as SP and GH. Besides LP, GH has the highest time complexity. However, the maximum execution time in our simulation is around 0.35 seconds, which is acceptable in network design.

5.7 Summary

Industrial WSANs face significant challenges in achieving long-term reliable communication in harsh environments. While the WirelessHART standard adopts graph routing to enhance network reliability, the problem of maximizing network lifetime for graph routing becomes a critical open problem. This chapter introduces and formulates the network lifetime maximization problem for graph routing. We present an optimal graph routing algorithm based on Integer Programming, and two efficient algorithms based on linear programming relaxation and greedy heuristics, respectively. We have implemented our graph routing algorithms on a physical WirelessHART network testbed. Experimental results on the testbed and in simulations show the linear relaxation and greedy heuristics can improve the network lifetime by up to 50% when compared to an existing graph routing algorithm. Moreover, the greedy heuristics requires significantly lower computation time, making it particularly suitable for WirelessHART networks that may compute graph routes frequently when facing network changes in open environments.

Chapter 6

Distributed Application Allocation in Shared Sensor Networks

Wireless sensor networks are evolving from single-application platforms towards an integrated infrastructure shared by multiple applications. Given the resource constraints of sensor nodes, it is important to optimize the allocation of applications to maximize the overall Quality of Monitoring (QoM). Recent solutions to this challenging application allocation problem are centralized in nature, limiting their scalability and robustness against network failures and dynamics. This chapter presents a distributed game-theoretic approach to application allocation in shared sensor networks. We first transform the optimal application allocation problem to a submodular game and then develop a decentralized algorithm that only employs localized interactions among neighboring nodes. We prove that the network can converge to a pure strategy Nash equilibrium with an approximation bound of 1/2. Simulations based on three real-world datasets demonstrate that our algorithm is competitive against a state-of-the-art centralized algorithm in terms of QoM.

6.1 Introduction

Traditionally, wireless sensor networks (WSNs) are used as specialized platforms where only a single application is deployed on each sensor. Recently, large-scale, integrated WSNs that support multiple applications start to emerge. Many application domains such as urban sensing [63], building automation and environmental monitoring [1] have already adopted the integrated WSN paradigm to support multiple applications. Compared to separate

application-specific sensor networks, a shared WSN can be more cost effective and more flexible as it enables resource sharing among applications and dynamic resource allocation in response to changes in the environment and user needs.

Severe resource constraints limit the allocation of all possible applications to sensors in a shared WSN. For example, the TelosB mote [2], a representative sensor platform, only has 10 KB of RAM, a 250 Kbps radio, and a 16-bit CPU running at 8 MHz. On the other hand, the Quality of Monitoring (QoM) of applications depends on application allocations. Therefore, it is important to optimize the allocation of multiple applications among sensor nodes in order to maximize the overall QoM, subject to resource constraints. This problem is challenging because it is essentially a discrete optimization with an exponentially large solution space.

Some recent works utilize the submodularity of the QoM function to tackle this discrete optimization problem. Submodularity is an important property of the QoM functions for networked sensing applications. Intuitively, a function f that maps a subset of a set S to a real value is submodular if it has a *diminishing return* property, i.e., adding an element to a smaller subset of S makes a bigger difference to the function values than adding it to a larger subset of S. The submodularity of QoM is due to the inherent property that sensor readings from different nodes are often correlated. For instance, since the temperature readings from different nodes in the same room are correlated with each other, allocating a new node to a temperature monitoring application results in diminishing improvement to the QoM as the set of nodes allocated to the application grows. Submodularity of sensor allocation for monitoring temperature [41] and water quality [90, 119, 120] has been observed in previous studies of real-world datasets.

Many existing works centered around submodular optimization have been proposed for optimization problems in sensor networks. Recent theoretical works also show approximation algorithms that can achieve a $(1-1/e)$-approximation bound [122]. Xu *et al.* [208] proposed a greedy algorithm and achieved a $1/3$ approximation bound. Submodular optimization approaches are also used in sensor selection and placement applications [119, 120]. However, all these existing submodular optimization approaches are essentially *centralized* solutions. For WSNs, a centralized algorithm implies there is either a node or gateway that maintains the global information of the network.

A centralized approach is not desirable for WSNs due to its limitations in scalability and fault tolerance. First, a shared WSN is usually of large scale in terms of the number of nodes and hop counts. Hence, it is inefficient or even impossible to achieve global information sharing that is required by centralized optimization algorithms. Second, in a centralized approach, much of the computation and communication happens on a single point resulting in a single point of failure. To address the limitations of centralized approaches, we study *distributed* optimization approaches for application allocations. Meanwhile, we still exploit the submodularity property of QoM functions to achieve desirable approximation bounds.

In this chapter, we provide several major theoretical results: 1) We propose the *covariance cover* function as a new QoM metric that is amenable to distributed optimization; 2) We show that the optimal multi-application allocation problem with covariance cover as objective function is a submodular optimization problem with multiple knapsack constraints; 3) We propose a game theory based distributed algorithm for solving this submodular optimization problem and prove that our algorithm can achieve a 1/2-approximation bound when each sensor achieves optimal allocation of applications. We also prove our algorithm can achieve a $1/(1+\beta)$-approximation bound when each sensor achieves a $1/\beta$-approximate allocation of applications. Simulations based on three real-world datesets demonstrate that our distributed algorithm can achieve comparable QoM as a state-of-the-art centralized algorithm [208], while scaling effectively in terms of both execution time per node and the communication overheads.

The rest of the chapter is organized as follows. Section 6.2 reviews the related works. Section 6.3 formulates the application allocation problem. Section 6.4 presents our distributed algorithm design. Section 6.5 presents approximation bounds of our algorithms. Section 6.6 evaluates our algorithm design in simulations. Section 6.7 concludes this chapter.

6.2 Related Works

Originated from centralized optimization, subgradient methods have been used to optimize problems where the gradients of the objectives are hard to obtain, while the subgradients of objective functions with respect to a subset of variables are easy to obtain [149, 158]. The subgradient optimization method can be used as a distributed optimization algorithm for

problems in WSNs, in which each sensor node optimizes the objective function distributively using its own subgradient value. However, the subgradient method is not suitable for the multi-application allocation problem in a large-scale, multi-hop WSN, due to the fact that in each iteration of the algorithm, it is still required to propagate the solution for the subsequent subgradient calculation. That is to say, although the optimization is localized to each sensor, global communication is still required.

Game-theoretic approaches have been proposed to address the above issue. In these approaches, communications are made only between certain sensors in a user-defined neighborhood. Another unique property of game-theoretic approaches is that they do not assume that agents (in this case, sensor nodes) work cooperatively. Instead, selfish sensor nodes optimize a local version of the objective functions, often called "utilities" or "private utilities", independently, until none of them can further improve their private utilities by making a different decision.[2] When these utilities are carefully designed to reflect the objective function, the overall objective function, also called "social utility", is subsequently optimized by these noncooperative agents [195].

When the social and private utilities are carefully designed, game-theoretic approaches guarantee a constant optimization bound [36,106,195]. Since the utility system decides the nature of the game, needless to say, for game-theoretic approaches to work for multi-application allocation problems, designing the utility system is critical. Specifically, in a distributed solution, a utility system that is easy to calculate and has no global information propagation requirements is desirable. In other words, in the application allocation problem, a utility system should reflect the QoM value based on the decisions of each sensor, while it does not require global communication in the network.

Previous works proposed different QoM formulations [40, 90, 208], including variance reduction and mutual information gain. However, neither is suitable as the objective function in a distributed game-theoretic approach, which requires that a sensor's utility is independent of other sensors that are not in its neighborhood. This condition is violated when using variance reduction or mutual information as QoM, because one sensor's utility of allocating an application is related to all sensors that carry the application. We address this issue by

[2]In Game Theory, a state where no player can improve its utility is called an equilibrium state.

proposing a new QoM metric that is submodular and suitable for game-theoretic distributed optimization while serving as an effective proxy for variance reduction in QoM optimization.

6.3 Problem Formulation

In this section, we first review the *variance reduction* QoM formulation. After discussing the disadvantages of using variance reduction in distributed algorithms, we propose a new QoM metric called *covariance cover* that is amenable to distributed solutions. In the end, we formulate the application allocation problem in shared WSNs using covariance cover as QoM metric.

6.3.1 QoM Formulation

Variance reduction is commonly used to measure QoM in WSN applications [90, 208]. Assuming sensor readings follow a Gaussian Process, the variance reduction measures how much the variance of the readings from the unallocated sensors.

Variance reduction is calculated based on covariance. Assuming K is the covariance matrix for sensor nodes, and for two subsets of sensor nodes $G, H \subseteq V$, the covariance matrix of G and H is denoted by K_{GH}, where its rows corresponding to G and columns corresponding to H extracted from K. For a given set G with application allocated, the variance of the unassigned set $\bar{G} = V \backslash G$ is

$$\sigma^2_{\bar{G}|G} = tr(K_{\bar{G}\bar{G}}) - tr(K_{\bar{G}G} K^{-1}_{GG} K_{G\bar{G}}),$$

where $tr()$ is the trace of a matrix.

In this application allocation problem, the goal is to minimize the variance of $\bar{G}$ given G such that the quality of sensing is maximized. Namely, we want to maximize the negation of the variance. Given $tr(K) = tr(K_{GG}) + tr(K_{\bar{G}\bar{G}})$, variance reduction for one application

is:

$$Q_{VR} = tr(K_{GG}) + tr(K_{\bar{G}G}K_{GG}^{-1}K_{G\bar{G}}) \tag{6.1}$$

Variance reduction is just one of the many possible ways to formulate QoM. There is an inherent disadvantage of using variance reduction for our problem. It is not feasible for a sensor node with limited memory resource to store a kernel matrix that is quadratic to the size of the network, and it is expensive to compute variance reduction since it involves matrix multiplication and inversion. To overcome this inherent disadvantage, we decompose the variance reduction and propose a new formulation which is more amenable to distributed approaches.

We begin with introducing the network model. A network consists of a group of sensors $\{1, 2, \cdots n\}$. Each sensor node can be presented as a vertex. For a pair of sensors i and j, if each of them is in the other's communication range, i and j are defined as a pair of neighbors. The network can be presented as a graph $G = (V, E)$, where $V = \{1, 2, \cdots, n\}$ and $(i, j) \in E$ if and only if i and j are a pair of neighbors. Now we will decompose the variance reduction based on two assumptions.

Theorem 2. *Variance reduction formulation (6.1) is equivalent to*

$$\sum_{i \in G} K_{ii} + \sum_{(i,j) \in E, i \in G \text{ or } j \in G} K_{ij}^2,$$

if (I) the covariance of any two nodes is nonzero if and only if they are a pair of neighbors, and (II) any two allocated nodes are not a pair of neighbors.

Proof: Let us first simplify the variance reduction formulation. Since any two allocated nodes are not neighbors of each other, and only neighbors have nonzero variance, we can prove K_{GG} is an identity matrix. It immediately follows that

$$\begin{aligned} Q_{VR} &= tr(K_{GG}) + tr(K_{\bar{G}G}K_{GG}^{-1}K_{G\bar{G}}) \\ &= \sum_{i \in G} K_{ii} + \sum_{i \in G, j \in \bar{G}} K_{ij}^2. \end{aligned}$$

Since only a pair of neighbors have nonzero covariance,

$$Q_{VR} = \sum_{i \in G} K_{ii} + \sum_{i \in G, j \in \bar{G}, (i,j) \in E} K_{ij}^2.$$

We assume two allocated nodes are not neighbors, which means if $(i,j) \in E, i \in G$, then $j \in \bar{G}$. It follows

$$Q_{VR} = \sum_{i \in G} K_{ii} + \sum_{(i,j) \in E, i \in G \text{ or } j \in G} K_{ij}^2. \quad \square$$

One question raises naturally: how realistic are the assumptions? We argue that the proposed two assumptions, although sometimes violated, provide good approximations of the real-world scenarios. First, it is reasonable to assume a pair of nearby sensors have larger covariance. For example, the temperature measurements of two different sensors in the same office room are more correlated than two sensors in different rooms. Second, since our applications have the inherent property of submodularity, allocating two neighboring nodes simultaneously typically does not give much gain in terms of QoM. To maximize QoM, a good solution should naturally allocate nodes that have unallocated nodes as neighbors. Actually, our submodular game algorithm is not limited by these two assumptions, it can handle situations when assumptions do not hold.

We name the new QoM formulation *covariance cover*. Denoting $\tau_{ij} = K_{ij}^2$ as the weight of edge (i,j), and $\tau_{ii} = K_{ii}$ as the weight of node i, we define the covariance cover formulation as

$$Q_{CC} = \sum_{i \in G} \tau_{ii} + \sum_{(i,j) \in E, i \in G \text{ or } j \in G} \tau_{ij}. \tag{6.2}$$

6.3.2 Application Allocation Problem Formulation

Given QoM metric as covariance cover, we want to further formulate the application allocation problem in shared sensor networks.

When there are multiple applications $P = \{1, 2, \cdots, p\}$ with weights $\{w^1, w^2, \cdots, w^p\}$, we want to maximize the summation QoM of all applications $\sum_{t=1}^{p} w^t Q^t$, where Q^t is the covariance cover for application t.

$$Q^t = \sum_{i \in G^t} \tau_{ii}^t + \sum_{(i,j) \in E, i \in G^t \text{ or } j \in G^t} \tau_{ij}^t$$

This problem is challenging because of critical resource constraints, e.g., CPU and memory constraints. For each sensor, the total memory and CPU consumed by all applications can not exceed its limits. Therefore, suppose each node has m resource constraints $R = \{1, 2, \cdots, m\}$, the capacity of node i on resource k is $C_{i,k}$, and application t consumes $c_{i,k}^t$ units of resource k on node i, the constrained optimization problem can be formulated as:

$$\begin{aligned} \max \quad & QoM = \sum_{t \in P} w^t Q^t \\ & Q^t = \sum_{i \in G^t} \tau_{ii}^t + \sum_{(i,j) \in E, i \in G^t \text{ or } j \in G^t} \tau_{ij}^t \\ \text{s.t.} \quad & \sum_{t | i \in G^t} c_{i,k}^t \leq C_{i,k}, \quad \forall i \in V, \forall k \in R \end{aligned}$$

here G^t is the set of nodes which are assigned application t.

It is easy to see that all resource constraints here are knapsack constraints. This type of constraint formulation also can be used to characterize various communication patterns among nodes, such as the pattern in a data collection application that collects data from every node on the routing tree.

6.4 Submodular Game

In this section, we will formulate a non-cooperative game based on the covariance cover formulation discussed in the previous section, which leads to a completely distributed algorithm. We introduce typical terminologies in game theory at first.

6.4.1 Submodular Game Formulation

Suppose we have n sensor nodes, and each sensor node i in the network is an agent i in the game. For each sensor, its strategy a_i is the subset of applications that can run on it.

$$\begin{aligned} a_i &= \{t| \text{ application } t \text{ runs on sensor } i\} \\ &= \{t|i \in G^t, \forall t \in P\}. \end{aligned}$$

Under the resource constraint we discussed earlier, the *strategy set* $\mathcal{A}_i$ of player i is

$$\mathcal{A}_i = \{a_i| \sum_{t \in a_i} c^t_{i,k} \leq C_{i,k}, \forall k \in R\},$$

A *pure strategy* is one in which each agent decides to carry out a specific strategy. In game theory, *mixed strategy* is also widely discussed. However, we only discuss pure strategy in this chapter, because in reality of sensor networks, it is hard to implement strategies with probability distribution. Also, we prove that our game has at least one Nash equilibrium with pure strategies. We denote the *strategy space* of the game as $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_n$.

A game is always defined on a utility system. To build the utility system, we need to define the utility function at first. Given a strategy profile $A = (a_1, a_2, \cdots, a_n) \in \mathcal{A}$, let $A \oplus a'_i$ denote the strategy profile obtained if agent i changes its strategy from a_i to a'_i. Formally, $A \oplus a'_i = (a_1, \cdots, a_{i-1}, a'_i, \cdots, a_n)$.

The goal of our game is to maximize the *social utility* $\gamma : 2^V \to \mathcal{R}$ defined on pure strategy profile $A = \{a_1, \cdots, a_n\}$ as

$$\begin{aligned} \gamma(A) &= \sum_{t=1}^{p} \gamma^t(A) \\ &= \sum_{t=1}^{p} w^t(\sum_{t \in a_i \text{ or } t \in a_j, (i,j) \in E} \tau^t_{ij} + \sum_{t \in a_i, i \in V} \tau^t_{ii}) \\ &= \sum_{t=1}^{p} w^t(\sum_{(i,j) \in E, i \in G^t \text{ or } j \in G^t} \tau^t_{ij} + \sum_{i \in G^t} \tau^t_{ii}). \end{aligned} \tag{6.3}$$

Remind G^t is the set of nodes who are assigned application t.

For each agent i, we define a *private utility* $\phi_i : 2^V \rightarrow \mathcal{R}$ as:

$$\begin{aligned}\phi_i(A) &= \sum_{t \in a_i} \phi_i^t(A) \\ &= \sum_{t \in a_i} w^t(\tau_{ii}^t + \sum_{j \in \mathcal{N}_i} \frac{\tau_{ij}^t}{1 + \delta_{j \in G^t}})\end{aligned} \tag{6.4}$$

where $\mathcal{N}_i = \{j | (i,j) \in E\}$ is sensor i's neighborhood. For edge (i,j), if not only i, but also j runs application t, (i,j)'s edge weight τ_{ij}^t need to be equally shared by both i and j. Otherwise, sensor i will account all (i,j)'s edge weight into its private utility.

The goal of each sensor is, therefore, to select a strategy in order to maximize its private utility under resource constraints. Clearly, such strategies may not produce a good solution with respect to the social utility γ. However, we will show that the strategies sensors finally select will result in a reasonable good social utility γ in next section.

To localize the optimization problem to each sensor, given strategies of its neighbors fixed, we redefine sensor i's private utility $\phi_i(A)$ as its utility function $u_i(x_i)$, which is a function of its own decisions x_i. Its decision $x_i = \{x_i^1, \cdots, x_i^p\}$ is redefined from its strategy a_i, where $x_i^t = 1$ means $t \in a_i$.

$$u_i(x_i) = \sum_{t=1}^{p} w^t [\tau_{ii}^t + \sum_{j \in \mathcal{N}_i} \frac{\tau_{ij}^t}{1 + \delta_{j \in G^t}}] x_i^t$$

We denote $\Omega_i^t = [\tau_{ii}^t + \sum_{j \in \mathcal{N}_i} \frac{\tau_{ij}^t}{1+\delta_{j \in G^t}}]$ as a constant, assuming strategies of i's neighbors are given. To maximize its utility function, sensor node i needs to solve a integer programming problem:

$$\begin{aligned} \text{Max} \quad & u_i(x_i) = \sum_{t=1}^{p} w^t \Omega_i^t x_i^t \\ \text{where} \quad & x_i^t \in \{0,1\}, \forall t \in P \\ \text{s. t.} \quad & \sum_{t=1}^{p} c_{i,k}^t \; x_i^t \leq C_{i,k} \;\; \forall k \in R \end{aligned} \tag{6.5}$$

Actually, this is a typical multidimensional knapsack problem. There is a rich package of literature to solve this problem. We propose two algorithms based on p, which is the number of applications. If p is not larger than T_p, our solution will adopt a naive enumeration algorithm. Basically, it enumerates all possible application assignments and returns an optimal solution. Otherwise, our solution will adopt a polynomial time approximate algorithm, which is proven to have a $\frac{1}{1+m}$ approximation bound (section 9.4.2 of [113]), where $m = |R|$ is the number of resource constraints. Here T_p is the threshold for p, we set it to 5 in our implementation. We show the sketch of our solution for problem (6.5) in Algorithm 7.

Algorithm 7: Algorithm for knapsack problem (6.5)

Set $\hat{x} = \{0, \cdots, 0\}$;
if $p \leq T_p$ **then**
    Adopt the Enumeration Algorithm;
    Enumerate $x \in \{0,1\}^p$, return optimal solution $\hat{x}$.
else
    Adopt the Approximation Algorithm;
    Relax problem (6.5) to a linear programming problem and compute an optimal solution x_{LP} of the LP-relaxation.
    Set $I = \{t | x^t_{LP} = 1\}$
    Set $F = \{t | 0 < x^t_{LP} < 1\}$
    Return $\hat{x} = \max\{\sum_{t \in I} w^t, \max\{w^t | t \in F\}\}$
end

Now we analyze computational cost of our algorithm. If $p \leq T_p$, the time complexity is $O(\binom{p}{d})$ where d is the maximum number of applications that can be allocated on one node. And if p is larger than T_p, the relaxed linear programming (LP) problem is significantly simplified due to the small numbers of resource constraints as well as applications. The number of resource constraints is usually no more than 3 (e.g., memory, CPU, and bandwidth). The number of applications is also small due to the limited resources available per node. Our algorithm employs an efficient and practical solution as follows. We solve the dual problem of the aforementioned LP problem which only has three variables (the shadow prices of memory, CPU and bandwidth constraints) and p constraints (p is the number of applications). Even a naive LP solver that enumerates all possible extreme points (each of the three constraints determines one extreme point) and finds the best feasible one has the computational complexity $O(p\binom{p}{3}) = O(p^4)$, and the memory requirement of the naive

enumeration algorithm is $O(1)$. Either way, the cost of each individual multidimentional knapsack problem is $O(p^d)$, where d is a small integer.

6.4.2 Submodular Game Algorithm

Now we discuss our distributed submodular game algorithm. In the beginning stage, sensor nodes in the network need to get two key parameters about applications set P: 1) types of required sensor readings; 2) the frequency of each sensor reading. These two key parameters are distributed to the network from a central facility like base station. After this stage, no central facility is needed in the algorithm, so our algorithm is fully distributed.

Algorithm 8: Game algorithm for sensor node i

initialization;

- i measures sensor readings for each *application* t;
- i broadcasts all sensor readings in its neighborhood;
- i calculates τ_{ii}^t and τ_{ij}^t for every neighbor j;

if *Timer* Λ_i *fires* **then**
  if *receiving strategy changes from neighbors* **then**
    i runs algorithm 7, output $\hat{x} \rightarrow$ strategy;
    if *strategy changes* **then**
      i broadcasts its strategy in neighborhood;
    end
  end
end

In the initialization stage, each node measures sensor readings for a certain interval and broadcasts sensor readings in its neighborhood. Based on neighbor j's readings, node i can calculate the covariance between i and j as well as τ_{ij}^t.

Algorithm 8 shows the detailed decision-making procedure for each sensor. In each round of the game, nodes share the same time interval T. Each node generates a random number Λ_i ($\Lambda_i < T$) as a timer using a unique seed, such that two timers will not fire at the same time. Each timer Λ_i will fire once and only once during each time interval T for sensor

node i to solve the allocation problem (6.5) locally. If a new strategy is generated, node i will broadcast it in the neighborhood. Otherwise i will keep quiet. Each sensor node i also receives messages from its neighbors about their updated strategies. The algorithm terminates when no strategy changes are made in a round.

Here we analyze the efficiency of our game algorithm. From the computational cost perspective, we already give the computational cost of each node in each round as $O(p^d)$. Since both p and d are small integers, it is reasonable to say the computational cost is acceptable on a sensor node with limited resources. From a network perspective, we want to analyze the communication cost. In each round, sensor node i needs to receive messages from all its neighbors and broadcast its own strategy in its neighborhood if necessary. We denote the *Expected Transmission Count (ETX)* of link e is ν_e. Since sensor node i broadcasts in the neighborhood, in the worst case, the number of messages it needs to send is the maximum of all the *ETXs* in its neighborhood $\zeta_i = \max_{j \in \mathcal{N}_i} \nu_{(i,j)}$. So the overall number of packets sensor i sends in the game is lower than $\kappa\zeta_i$, given κ is the number of overall number of rounds. Because κ is always a small number (less than 12) based on our evaluation, the communication cost is relatively small.

6.5 Convergence and Approximation Bound

In this section, we first show the social utility (6.3) is submodular. Then we prove our submodular game $(\gamma, \cup_{i \in V} \phi_i)$ defined in (6.3) and (6.4) can converge to a pure strategy Nash equilibrium with an approximation bound of $\frac{1}{2}$, if sensors use the enumeration algorithm to solve the multidimensional knapsack problem (6.5). If sensors use the $\frac{1}{1+m}$-approximate solution for the knapsack problem, the game can converge to a $(1+m)$-approximate pure strategy Nash equilibrium and the approximation bound is $\frac{1}{2+m}$, where m is number of resource constraints.

6.5.1 Submodularity

Definition 4. ***(Submodularity)*** *Let V be a finite set, a function $f : 2^V \to R$ is* submodular *if for any $A \subseteq B \subseteq V$ and $x \in V - B$, $f(B \cup \{x\}) \leq f(A \cup \{x\})$.*

Recall that we defined the social utility (6.3) as a function of pure strategy profiles in the last section. We redefine the social utility here as a function of the set of sensors that we allocate applications on: $\gamma = \sum_{t=1}^{p} w^t Q^t(G^t)$, where

$$Q^t(G^t) = \sum_{\{(i,j)|i \in G^t \text{ or } j \in G^t\}} \tau_{ij}^t + \sum_{i \in G^t} \tau_{ii}^t.$$

This definition is equivalent to the one we defined in (6.3), but it is now defined on the set of sensors. Based on this set based definition, we can prove the social utility γ is submodular.

Theorem 3. *The social utility γ is submodular.*

Proof: Since $\gamma = \sum_{t=1}^{p} w^t Q^t(G^t)$, we only need to show $Q^t(G^t), \forall t \in P$ is a submodular function. By definition, we need to prove: if $A \subseteq B \subseteq V$ and $x \in V - B$, $f(B \cup \{x\}) \leq f(A \cup \{x\})$.

If $x \in B$, it is obvious that $Q^t(B \cup \{x\}) - Q^t(B) = 0 \leq Q^t(A \cup \{x\}) - Q^t(A)$.

If $x \notin B$, it follows:

$$\begin{aligned}
& A \subseteq B \\
\Rightarrow \quad & \{(i,k)|i = x \,\&\, k \notin B\} \subseteq \{(i,k)|i = x \,\&\, k \notin A\} \\
\Rightarrow \quad & \sum_{i=x \,\&\, k \notin B} \tau_{ik}^t \leq \sum_{i=x \,\&\, k \notin A} \tau_{ik}^t \\
\Rightarrow \quad & Q^t(B \cup \{x\}) - Q^t(B) \leq Q^t(A \cup \{x\}) - Q^t(A) \quad \square
\end{aligned}$$

6.5.2 Convergence and Pure Nash Equilibrium

Now we will discuss the convergence of our game. By defining a potential function, we show the increase of each agent's private utility will lead to the increase of the potential function. Then we can prove our game will converge at a *pure strategy Nash equilibrium*. Here we assume our *Submodular Game Algorithm* (Algorithm 8) is using the enumeration algorithm.

Definition 5. ***(Pure Strategy Nash Equilibrium)*** *A pure strategy profile $A \in \mathcal{A}$ is a* pure strategy Nash equilibrium *if no agent has an incentive to change its strategy. For any agent i,*

$$\forall a_i' \in \mathcal{A}_i, \quad \phi_i(A \oplus a_i') \leq \phi_i(A).$$

Equivalently, given the other agents' strategies, a_i is the best response of agent i.

Theorem 4. *A pure strategy Nash equilibrium always exists for the utility system $(\gamma, \cup_i \phi_i)$ we defined in (6.3) and (6.4). And Submodular Game Algorithm (Algorithm 8) converges to a pure strategy Nash equilibrium.*

Proof: The proof starts from defining the potential function of the game. We define the potential function ψ for a strategy profile A as

$$\psi(A) = \sum_{t=1}^{p} w^t \Big(\sum_{i \in V, t \in a_i} \tau_{ii}^t + \sum_{(i,j) \in E, t \in a_i \text{ or } t \in a_j} \sum_{l=1}^{n_{(i,j)}^t} \frac{\tau_{ij}^t}{l} \Big)$$

where $n_{(i,j)}^t$ is the number of agents which are assigned application t as well as the end points of edge (i,j). $n_{(i,j)}^t$ is 2 if both i and j are assigned application t, and it is 1 if only one of i and j is assigned the application t.

Assume sensor i changes its strategy from a_i to a_i', as a result the strategy profile of the game changes from A to A'. Here a_i is the set of applications which are assigned to sensor i in original strategy profile A, and a_i' is that in new strategy profile A'. Let $G = a_i - a_i'$ and $H = a_i' - a_i$. We use E_i to denote the set of edges which coincide with sensor i. Since the change only happens on node i and edges sit on i, we will ignore other nodes and edges in following proof.

$$
\begin{aligned}
& \psi(A') - \psi(A) \\
= \quad & \sum_{t\in a_i'} w^t \tau_{ii}^t - \sum_{t\in a_i} w^t \tau_{ii}^t + \\
& \sum_{j\in\mathcal{N}_i}\sum_{t\in H} w^t \frac{\tau_{ij}^t}{n_{(i,j)}^t + 1} - \sum_{j\in\mathcal{N}_i}\sum_{t\in G} w^t \frac{\tau_{ij}^t}{n_{(i,j)}^t}; \\
& \phi_i(A') - \phi_i(A) \\
= \quad & \sum_{t\in a_i'} w^t \tau_{ii}^t - \sum_{t\in a_i} w^t \tau_{ii}^t + \\
& \sum_{t\in H} w^t \sum_{j\in\mathcal{N}_i} \frac{\tau_{ij}^t}{n_{(i,j)}^t + 1} - \sum_{t\in G} w^t \sum_{j\in\mathcal{N}_i} \frac{\tau_{ij}^t}{n_{(i,j)}^t}
\end{aligned}
$$

Obviously, $\psi(A') - \psi(A) = \phi_i(A') - \phi(A)$, we prove that the increase of the private utility of i is exactly the same as increase of the potential function of the game.

Once each individual sensor improves its private utility, the potential function ψ of the game also gets increased. Since the maximum value of this potential function is finite, the algorithm will converge in finite rounds. □

6.5.3 Valid Utility Game and Approximate Nash Equilibrium

Now we want to prove our submodular game $(\gamma, \cup_{i\in V}\phi_i)$ is a *valid utility system.*

Definition 6. ***(Utility System)*** *[195] A game is called a* utility system *if and only if the private utility of an agent is at least as great as the loss in social utility resulting from the agent dropping out of the game. That is, the game $(\gamma, \cup_i\phi_i)$ is a* utility system *if and only if it has the property $\phi_i(A) \geq \gamma'_{a_i}(A \oplus \emptyset_i)$.*

Definition 7. ***(Valid Utility System)*** *[195] A utility system is said to be* valid *if and only if the sum of private utilities of the agents is at most the social utility. That is, the utility system $(\gamma, \cup_i\phi_i)$ is a* valid utility system *if and only if it has the property $\sum_i^n \phi_i(A) \leq \gamma(A)$.*

We want to prove that the game we defined in (6.3) and (6.4) is a valid utility system. At first, we prove it is a utility system.

Theorem 5. *The game $(\gamma, \cup_i\phi_i)$ defined in (6.3) and (6.4) is a utility system.*

Proof:

$$\begin{aligned}
& \gamma'_{a_i}(A \oplus \emptyset_i) \\
= \quad & \gamma(A) - \gamma(A^{-i} \oplus \emptyset_i) \\
= \quad & \sum_{t \in a_i} w^t \Big(\sum_{j \in \mathcal{N}_i | t \notin a_j} \tau^t_{ij} + \tau^t_{ii} \Big) \\
\leq \quad & \sum_{t \in a_i} w^t \Big(\sum_{j \in \mathcal{N}_i | t \notin a_j} \tau^t_{ij} + \sum_{j \in \mathcal{N}_i | t \in a_j} \frac{\tau^t_{ij}}{2} + \tau^t_{ii} \Big) \\
= \quad & \phi_i(a_i). \qquad \square
\end{aligned}$$

Theorem 6. *The utility system $(\gamma, \cup_i \phi_i)$ defined in (6.3) and (6.4) is valid.*

Proof: We need to prove the utility system $(\gamma, \cup_i \phi_i)$ has the property $\sum_{i \in V} \phi_i(A) \leq \gamma(A)$.

First, we define the set of covered edges for application t as $E^t = \{(i,j) | i \in G^t \text{ or } j \in G^t\}$. Equation (6.3) shows that each $e = (i,j) \in E^t$ contributes τ^t_{ij} to $\gamma(A)$. We use a vector (ξ_i, ξ_j) to denote e's contribution to $\phi_i(A)$ and $\phi_j(A)$. There are three cases here:

$$(\xi_i, \xi_j) = \begin{cases} (\tau^t_{ij}, 0), & \text{if } i \in G^t, j \notin G^t \\ (0, \tau^t_{ij}), & \text{if } i \notin G^t, j \in G^t \\ (\frac{1}{2}\tau^t_{ij}, \frac{1}{2}\tau^t_{ij}), & \text{if } i \in G^t, j \in G^t \end{cases}$$

Since e's contribution to $\gamma(A)$ equals to the sum of its contribution to $\phi_i(A)$ and $\phi_j(A)$, after we sum up all $e \in E$,

$$\sum_{t \in P} w^t \sum_{(i,j) \in E, i \in G^t \text{ or } j \in G^t} \tau^t_{ij} = \sum_{i \in V} \Big(\sum_{t \in a_i} w^t \sum_{j \in \mathcal{N}_i} \frac{\tau^t_{ij}}{n^t_e} \Big).$$

Now we consider contribution of nodes, it is obviously that

$$\sum_{t \in P} w^t \sum_{i \in G^t} \tau^t_{ii} = \sum_{i \in V} \sum_{t \in a_i} w^t \tau^t_{ii}.$$

Combining both contribution of edges and nodes,

$$\gamma(A) = \sum_{i \in V} \phi_i(A). \qquad \square$$

We cite below an important result on valid game [195].

Lemma 5. *Let γ be a non-decreasing, submodular set function. If $(\gamma, \cup_i \phi_i)$ is a valid utility system then for any pure strategy Nash equilibrium $A^* \in \mathcal{A}$, we have $\gamma(A^*) \geq \frac{1}{2} OPT$, where OPT is the optimal social utility.*

Combining Theorem 3, Theorem 4, Theorem 6 and Lemma 5, we get following theorem.

Theorem 7. *For the submodular game $(\gamma, \cup_i \phi_i)$ we defined in (6.3) and (6.4), there exists at least one pure strategy Nash equilibrium. And for its any pure strategy Nash equilibrium $A^* \in \mathcal{A}$, we have $\gamma(A^*) \geq \frac{1}{2} OPT$.*

Now we consider the case in which each sensor runs the approximation algorithm and can only get a $\frac{1}{\beta}$ approximate solution instead of optimal solution for the multidiminutional knapsack problem. $\frac{1}{\beta}$ approximate solution ($\beta > 1$) means the solution is not less than $\frac{1}{\beta}$ of the optimal solution. We can prove that our algorithm can achieve a *β-approximate Nash Equilibrium.*

Definition 8. ***(β-approximate Nash Equilibrium)*** *A pure strategy profile $A \in \mathcal{A}$ is a β-approximate Nash equilibrium if no agent can find a better alternative pure strategy in which its private utility is more that β times better than its current private utility. That is for any agent i,*

$$\forall a_i' \in \mathcal{A}_i, \quad \phi_i(A \oplus a_i') \leq (1 + \beta)\phi_i(A)$$

Theorem 8. *For the submodular game defined in (6.3) and (6.4), Submodular Game Algorithm (Algorithm 8) with the $\frac{1}{\beta}$-approximation algorithm converges to a β-approximate Nash equilibrium $A \in \mathcal{A}$.*

Proof: The proof follows the same way of theorem 4. By bounding the value of the potential function, we can prove out Submodular Game Algorithm can reach a β-approximate Nash equilibrium. □

We cite the following important result on approximate Nash equilibria [195].

Lemma 6. *Let γ be a non-decreasing, submodular set function, and $(\gamma, \cup_i \phi_i)$ be a valid utility system. In any β-approximate Nash equilibrium $A \in \mathcal{A}$ we have $\gamma(A) \geq \frac{1}{1+\beta} OPT$, where OPT is the optimal social utility.*

Theorem 9. *For the submodular game defined in (6.3) and (6.4), Submodular Game Algorithm (Algorithm 8) with a β-approximate solution converges to a β-approximate Nash equilibrium $A \in \mathcal{A}$, and we have $\gamma(A) \geq \frac{1}{1+\beta} OPT$, where OPT is the optimal social utility.*

Theorem 9 follows Theorem 6, Theorem 8 and Lemma 6.

In our implementation, we use a $\frac{1}{1+m}$-approximation algorithm, so our Submodular Game Algorithm can converge to a $(1+m)$-approximate Nash equilibrium with an $\frac{1}{2+m}$ approximation bound.

6.6 Evaluation

In this section, we evaluate our **Submodular Game** Algorithm **(SG)** by comparing it against a state-of-the-art centralized optimization algorithm **Fractional Relaxation Greedy (FRG)** [208]. We conduct simulations on three real world datasets:

Intel dataset is collected in Intel Berkley lab [3]. 54 Mica2Dot sensor nodes with weather boards were used to collect topology information along with humidity, temperature and light values. The data collection last for more than one month at a sampling period of 31 seconds. In our evaluation, we generate the covariance matrices using data collected from 20 nodes in one day.

DARPA dataset is collected in the DARPA SensIT vehicle detection experiments [69]. 75 WINS NG 2.0 nodes are deployed to detect vehicles driving through several intersecting

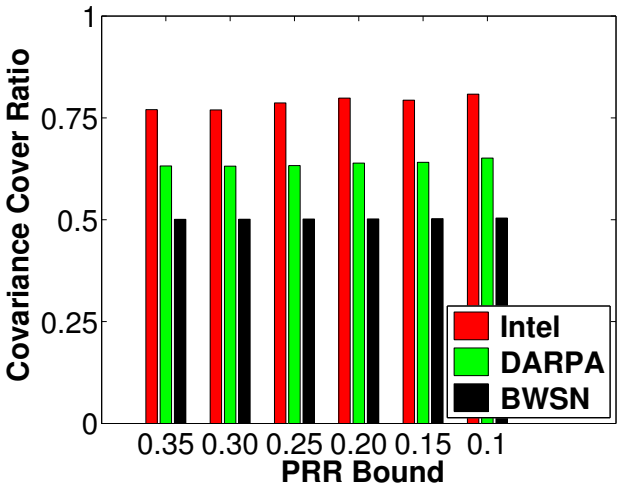


Figure 6.1: Covariance Cover Ratio

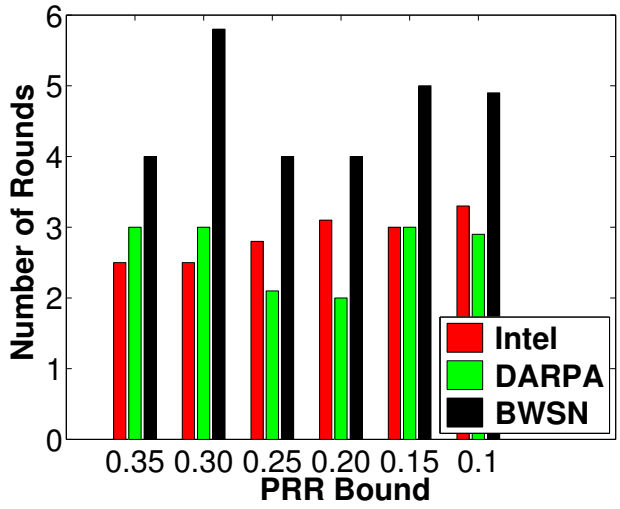


Figure 6.2: Number of Rounds

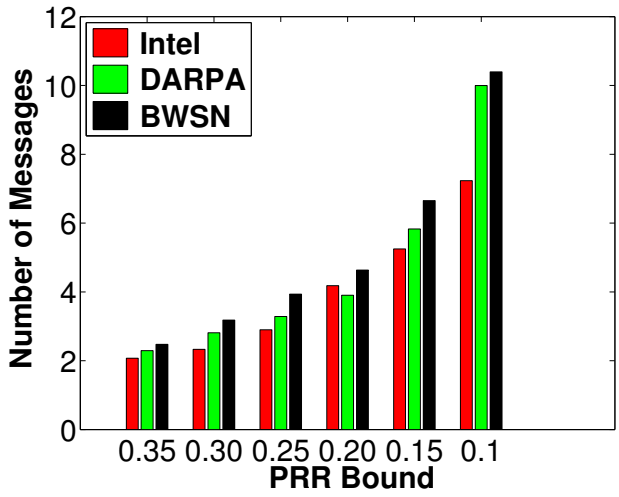
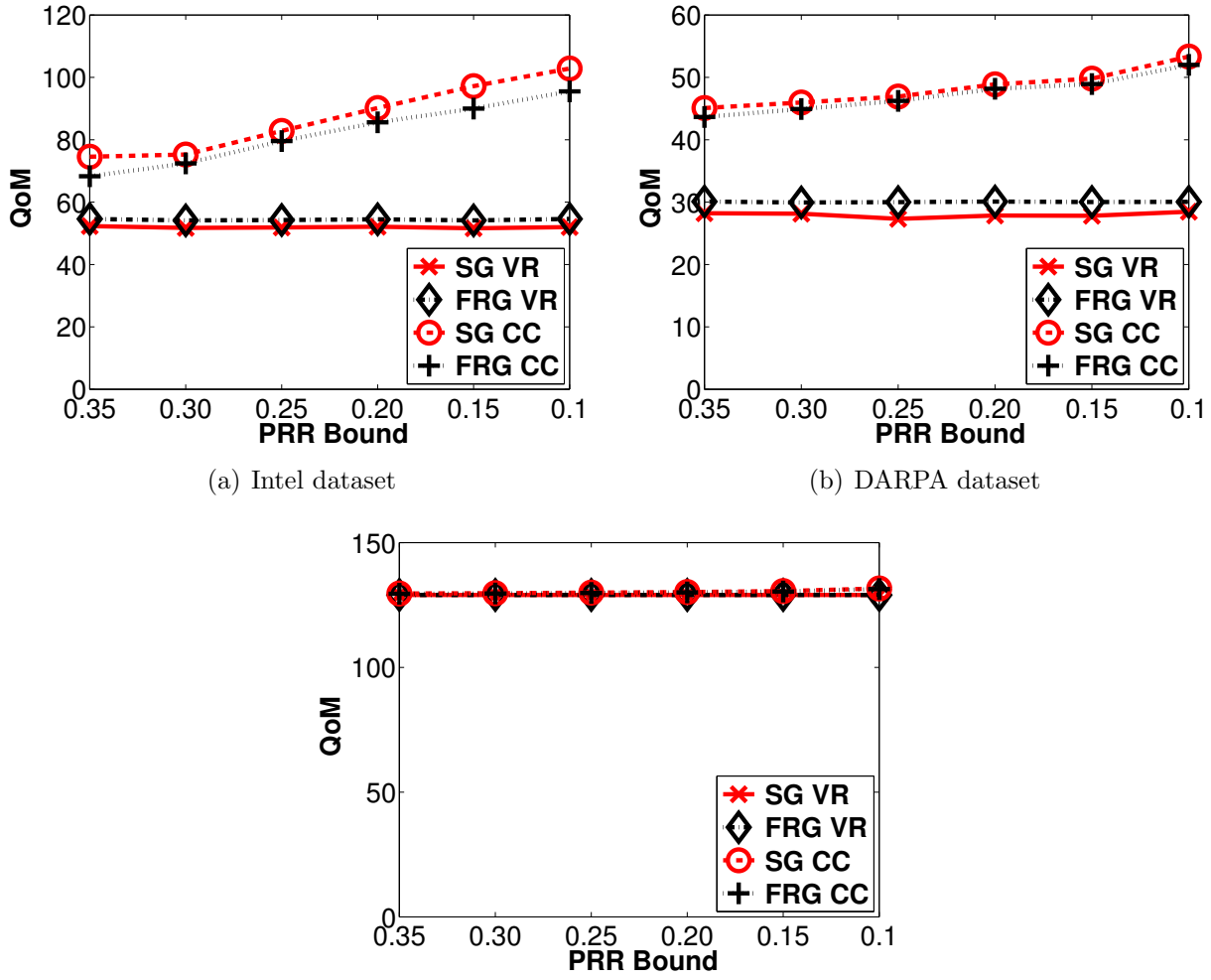


Figure 6.3: Number of messages per node

roads in 29 Palms, CA. Each WINS NG 2.0 node is equipped with three sensing modalities: acoustic (microphone), seismic (geophone) and infrared (polarized IR sensor). All nodes are deployed in an area of approximately $900 \times 300m^2$. In our evaluation, we use acoustic and seismic readings from 23 nodes in the dataset to generate covariance matrices.

BWSN dataset is acquired by running simulations on a 129-node sensor network used in Battle of the Water Sensor Networks (BWSN) [152]. We use the "bwsn-utilities" [4] program to simulate 10000 random injection events to this network for a duration of 96 hours and use the generated event detection data to calculate the covariance matrices. We use two event injection strategies to build two sets of data as two applications.

For each dataset, we can calculate the covariance matrices based on the sensor readings. The Packet Reception Ratio (PRR) of each link is included in the Intel dataset. We generate PRR for DARPAR and BWSN datasets based on location information of sensors following the way proposed in [222]. We then generate different network topologies by assigning different PRR bounds. Only links with PRR higher than the PRR bound is used for communication. In our simulations, we repeat Algorithm 8 in Section 6.4 10 times for each network topology. Because the number of applications is at most 3 in our simulations, we employ naive enumeration to solve the multidimensional knapsack problem (6.5) on each sensor node. As we proved in Theorem 7, SG will terminate at a pure strategy Nash equilibrium and the approximation bound is no less than $\frac{1}{2}$. We implement our SG algorithm in Matlab. All



(a) Intel dataset

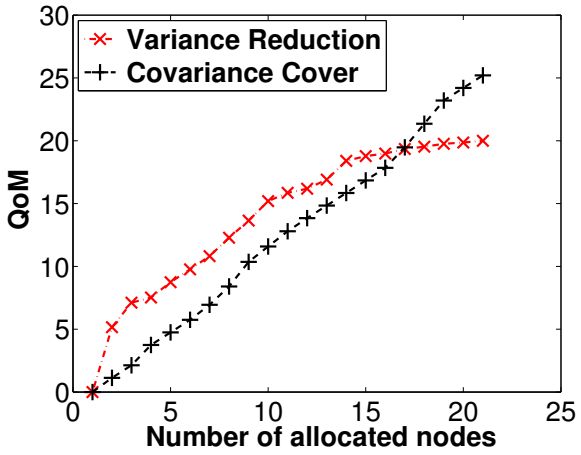
(b) DARPA dataset

(c) BWSN dataset

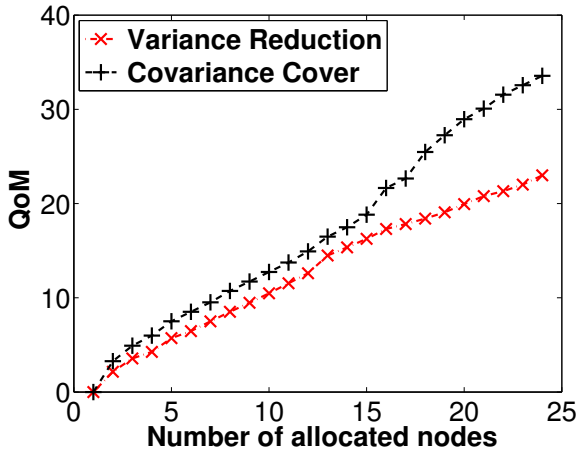
Figure 6.4: QoM Performance Analysis

results are gathered on a Macbook Pro machine with CPU frequency at 2.4GHz and 4GB memory.

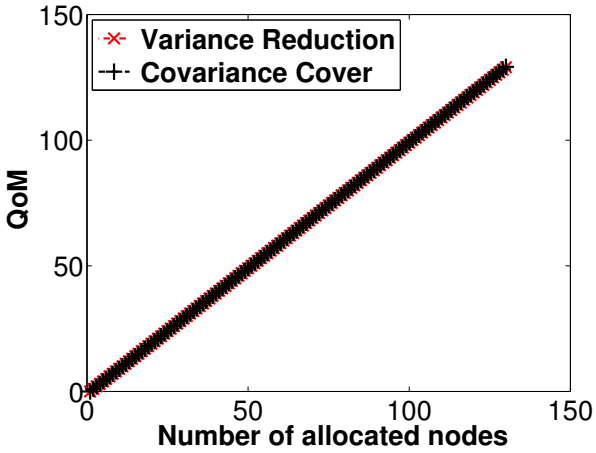
We define *covariance cover ratio* as the ratio between covariance cover achieved by the algorithm and the maximum covariance cover in the network. Since searching an optimal solution is too computational expensive, to assess the tightness of our bound, we compare our solution with the maximum covariance cover, i.e., the sum of all edge weights and node weights in the network. Figure 6.1 shows that the covariance cover of our solution is consistently no less than half of the maximum covariance cover, which means our solution



(a) Intel dataset



(b) DARPA dataset



(c) BWSN dataset

Figure 6.5: Comparison between VR and CC

is no less than half of the optimal solution. Results in Figure 6.1 indicates the tightness of our $\frac{1}{2}$ bound.

Figure 6.2 shows the maximum number of rounds for SG to converge is below 6 across all cases in all three real-world data sets. The communication cost of our algorithm is evaluated in Figure 6.3. It shows the average number of messages sent per node. As each sensor node has more neighbors with a lower PRR threshold, the average number of messages sent by each node increases from 2 to 10. Our results show that the communication cost required by SG in terms of number of packets is moderate.

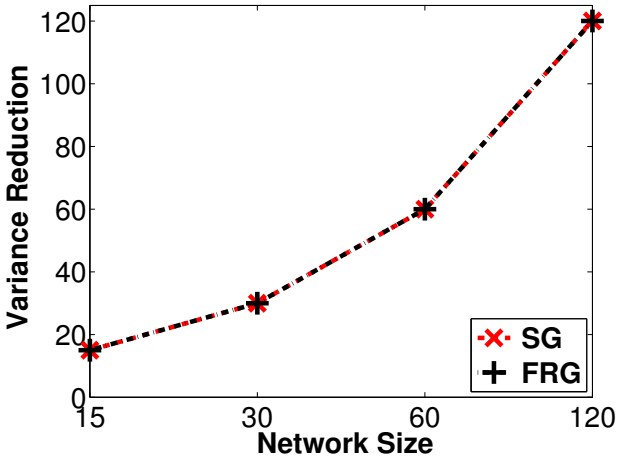


Figure 6.6: Variance Reduction

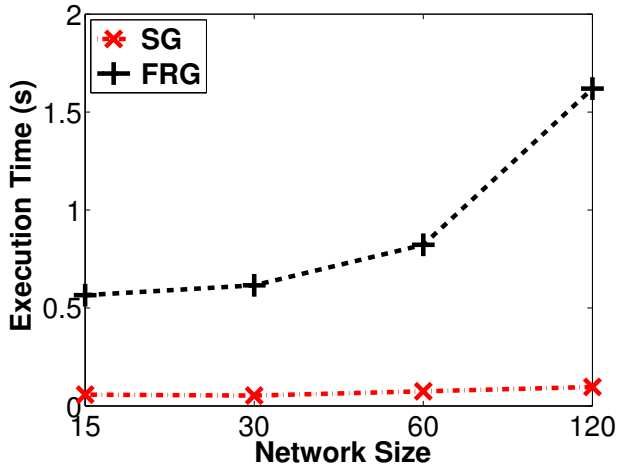


Figure 6.7: Execution Time

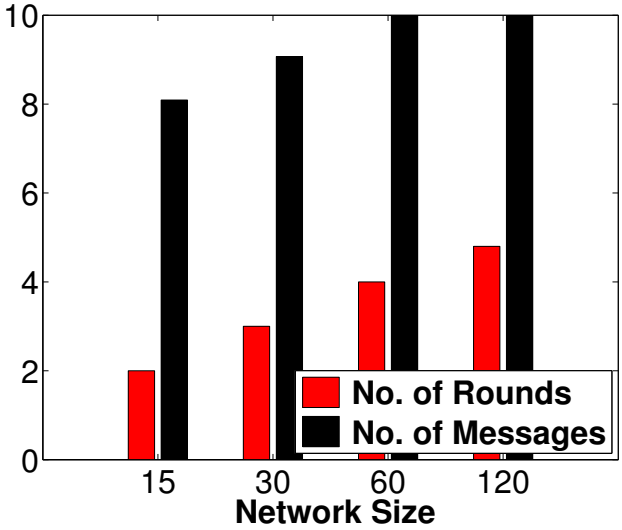


Figure 6.8: Cost Analysis

Figure 6.4 compares the performance of SG with FRG [208] in terms of variance reduction (VR) and covariance cover (CC). The variance reduction delivered by SG is always above 98% of that achieved by FRG in all three datasets. The covariance cover of SG is consistently higher than FRG. For BWSN dataset, the difference between different methods is within 2, which makes four curves difficult to tell. This result indicates the decentralized approach employed by SG is competitive with the centralized solution in terms of QoM.

Figure 6.5 investigates the correlation between covariance cover and variance reduction. We increase the number of allocated sensor nodes ϱ under same PRR threshold 0.5. It is difficult to distinguish VR and CC in BWSN dataset, because the difference of them is within 1. Results in the other two datasets show variance reduction and covariance cover are very close when ϱ is less than $n/2$, where n is total number of sensors. This is because when ϱ is small, allocated nodes are not neighbors of each other, which coincides with our assumption in Theorem 2. The difference increases when ϱ exceeds $n/2$, but a higher covariance cover is always associated with a higher variance reduction. This result shows that covariance cover can be used as an effective proxy to optimize the variance reduction of a node allocation.

We evaluate the scalability of our algorithm by selecting different subsets of sensor nodes from the BWSN dataset. Figure 6.6 shows SG is highly competitive against FRG in terms of variance reduction. Note the difference between SG and FRG is consistently within 2,

hence the SG and FRG curves are almost indistinguishable here. The execution times of SG and FRG for varying size of networks are compared in Figure 6.7. Since the SG is a distributed algorithm, we show the average execution time per node. For FRG, we show its overall execution time because it is a centralized algorithm. Our results show that SG remains fast as the number of nodes increases, with the run time remaining below 0.1 second. While the Macbook machine used in our simulation is more powerful than typical sensor nodes, the short execution times nevertheless indicates that SG is practical on sensors. More importantly, the solution scales effectively with network size. In comparison, the run time of FRG increases significantly as the number of nodes increases.

It is important to note that SG brings significant advantages than a centralized algorithm in several important ways. It does not incur the communication overhead for collecting the topology information of the entire network. Furthermore, it is robust against network disconnection as it does not depend on a single base station.

In Figure 6.8, we analyze the number of rounds and communication cost of SG. Both the number of rounds and messages per node increase moderately as the network size increases. The number of rounds remains within 10, indicating the scalability of our decentralized algorithm.

6.7 Summary

This chapter presents a distributed game-theoretic approach to application allocation in shared sensor networks. We first transform the optimal application allocation problem to a submodular game and then develop a decentralized algorithm that only employs localized interactions among neighboring nodes. We prove that the network can converge to pure strategy Nash equilibrium with a approximation bound. Simulations based on three real-world datasets demonstrate that our algorithm is competitive against a state-of-the-art centralized algorithm while scaling effectively with network size.

Chapter 7

Conclusion

With the emergence of industrial standards such as WirelessHART [26] and ISA100 [16], process industries are adopting *Wireless Sensor-Actuator Networks (WSANs)* that enable sensors and actuators to communicate through low-power wireless mesh networks [184]. Industrial process control applications impose stringent end-to-end latency requirements on data communication. To support a feedback control loop, the network periodically delivers data from sensors to a controller and then delivers its control input data to the actuators within an end-to-end deadline. Consequences of deadline misses in data communication may range from production inefficiency, equipment destruction to irreparable financial and environmental impacts.

To meet the stringent real-time performance requirements of control systems, there is a critical need for fast end-to-end delay analysis for real-time flows that can be used for online admission control. We present a new end-to-end delay analysis for periodic flows whose transmissions are scheduled based on the Earliest Deadline First (EDF) policy. Our analysis comprises novel techniques to bound the communication delays caused by channel contention and transmission conflicts in a WSAN. Furthermore, we propose a technique to reduce the pessimism in admission control by iteratively tightening the delay bounds for flows with short deadlines. Experiments on a WSAN testbed and simulations demonstrate the effectiveness of our analysis for online admission control of real-time flows.

Routing has significant impacts on reliability, real-time capacity and network lifetime. The core contributions of this dissertation tackles the real-time communication and network lifetime problems in WSAN routing. We first design real-time routing algorithms that leverage the insights from the delay analysis. By incorporating conflict delays in the routing decisions,

our real-time routing algorithms allow WSANs to accommodate more feedback control loops while meeting their deadline constraints.

Our second contribution to routing addresses the energy constraints of field devices in WSANs. Since many industrial devices operate on batteries in harsh environments where changing batteries are labor-intensive, WSANs need to achieve long network lifetime. To meet industrial demand for long-term reliable communication, we propose efficient graph routing designs to maximize network lifetime of WSANs. We first formally formulate the network lifetime maximization problem for WSANs under graph routing and prove it is NP-complete. We then propose the optimal algorithm and two more efficient algorithms to prolong the network lifetime of WSANs. Experiments in a physical testbed and simulations show our linear programming relaxation and greedy heuristics can improve the network lifetime by up to 50% while preserving the reliability benefits of graph routing.

Besides industrial WSANs, we have seen wireless sensor networks built as an integrated infrastructure shared by multiple environmental monitoring applications. Given the resource constraints of sensor devices, it is important to optimize the allocation of applications to maximize the overall quality of sensing. Recent solutions to this challenging application allocation problem are centralized in nature, limiting their scalability and robustness against network failures and dynamics. We present a distributed game-theoretic approach to allocate monitoring applications. We first transform the application allocation problem to a submodular game and then develop a decentralized algorithm that only employs localized interactions among neighboring devices. We prove that the network can converge to a pure strategy Nash equilibrium with an approximation bound of 1/2. Simulations based on three real-world datasets demonstrate that our algorithm is competitive against a state-of-the-art centralized algorithm in terms of quality of sensing.

References

[1] http://research.cens.ucla.edu/areas/2005/NIMS/.

[2] http://www.memsic.com/products/wireless-sensor-networks/wireless-modules.html.

[3] http://db.csail.mit.edu/labdata/labdata.html.

[4] http://www.water-simulation.com/wsp/about/bwsn/.

[5] http://www2.emersonprocess.com/siteadmincenter/PM

[6] http://www.reuters.com/article/idUSN1124632920100512.

[7] http://www.hse.gov.uk/pubns/regindex.htm.

[8] http://wsn.cse.wustl.edu/index.php/Testbed.

[9] http://www.tinyos.net/.

[10] Bluetooth. http://www.bluetooth.com.

[11] CC2420 documentation. http://www.ti.com/lit/ds/symlink/cc2420.pdf.

[12] Control with WirelessHART. http://www.hartcomm.org/protocol/training/resources/wiHART_resources/Control_with_WirelessHART.pdf.

[13] Emerson's WirelessHART report. http://www2.emersonprocess.com/en-us/plantweb/wireless/pages/wirelesshomepage-flash.aspx.

[14] GNU Linear Programming Kit. http://www.gnu.org/software/glpk/.

[15] HART Communication. http://www.hartcomm2.org/index.html.

[16] ISA100: Wireless Systems for Automation. https://www.isa.org/isa100/.

[17] Kronecker delta function. http://en.wikipedia.org/wiki/Kronecker_delta.

[18] The MOSEK optimization tools manual. http://docs.mosek.com/6.0/tools/node007.html.

[19] Neos server. http://www.neos-server.org/neos/.

[20] Technical overview of time synchronized mesh protocol (TSMP). Technical report, DUST Networks. http://www.dustnetworks.com/cms/sites/default/files/TSMP_Whitepaper.pdf.

[21] WINA. http://www.wina.org.

[22] Wirelesshart adapter technical specification. http://www.linear.com/product/LTP5901-WHM.

[23] Wirelesshart network topology. http://en.hartcomm.org/hcp/tech/wihart/wireless_how_it_works.html.

[24] WiSA: Wireless sensor and actuator networks for measurement and control. http://www.control.hut.fi/Research/wisa.

[25] ZigBee alliance. http://www.zigbee.org.

[26] WirelessHART specification, 2007. http://www.hartcomm2.org.

[27] Tarek F. Abdelzaher, Shashi Prabh, and Raghu Kiran. On real-time capacity limits of multihop wireless sensor networks. In *RTSS '04*.

[28] Micah Adler, Arnold L. Rosenberg, Ramesh K. Sitaraman, Walter Unger, and Lehrstuhl Fur Informatik I. Scheduling time-constrained communication in linear networks. In *In Proc. 10th Ann. ACM Symp. on Parallel Algorithms and Architectures*, pages 269–278, 1998.

[29] Gahng-Seop Ahn, A.T. Campbell, A. Veres, and Li-Hsiang Sun. Swan: service differentiation in stateless wireless ad hoc networks. In *INFOCOM '02: Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 457–466, 2002.

[30] Matthew Andrews and Lisa Zhang. Routing and scheduling in multihop wireless networks with time-varying channels. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1031–1040, 2004.

[31] V. Annamalai, S. K. S. Gupta, and L. Schwiebert. On tree-based convergecasting in wireless sensor networks. 2003.

[32] T.P. Baker. Multiprocessor edf and deadline monotonic schedulability analysis. In *RTSS'03*.

[33] T.P. Baker. An analysis of edf schedulability on a multiprocessor. *IEEE Transactions on Parallel and Distributed Systems*, 16(8):760 – 768, aug. 2005.

[34] Sanjoy Baruah. Techniques for multiprocessor global schedulability analysis. In *RTSS'07*.

[35] Y. Bejerano, Dongwook Lee, P. Sinha, and L. Zhang. Approximation algorithms for scheduling real-time multicast flows in wireless lans. In *INFOCOM '08: The 27th IEEE Conference on Computer Communications*, pages 2092–2100, April 2008.

[36] Oren Ben-zwi and Amir Ronen. The local and global price of anarchy of graphical games. In *SAGT*, 2008.

[37] M Bertogna and M Cirinei. Response-time analysis for globally scheduled symmetric multiprocessor platforms. In *RTSS '07*.

[38] Marko Bertogna, Michele Cirinei, and Giuseppe Lipari. Improved schedulability analysis of edf on multiprocessor platforms. In *ECRTS'05*.

[39] Marko Bertogna, Michele Cirinei, and Giuseppe Lipari. Schedulability analysis of global scheduling algorithms on multiprocessor platforms. *IEEE TPDS*, 20(4):553 – 566, April 2009.

[40] Sangeeta Bhattacharya, Abusayeed Saifullah, Chenyang Lu, and Grui Catalin Roman. Multi-application deployment in shared sensor networks based on quality of monitoring. In *RATS*, 2010.

[41] Fang Bian, David Kempe, and Ramesh Govindan. Utility-based sensor selection. In *IPSN*, 2006.

[42] T. Blevins, G. McMillan, W. Wojsznis, and M. Brown. Advanced control unleashed: Plant performance management for optimum benefit. *ISA Press*, 2002.

[43] Gurashish Brar, Douglas M. Blough, and Paolo Santi. Computationally efficient scheduling with the physical interference model for throughput improvement in wireless mesh networks. In *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*, pages 2–13. ACM, 2006.

[44] B.D. Bui, R. Pellizzoni, M. Caccamo, C.F. Cheah, and A. Tzakis. Soft real-time chains for multi-hop wireless ad-hoc networks. In *RTAS '07*.

[45] Marco Caccamo and Lynn Y. Zhang. The Capacity of Implicit EDF in Wireless Sensor Networks. In *ECRTS'03*.

[46] Marco Caccamo, Lynn Y. Zhang, Lui Sha, and Giorgio Buttazzo. An Implicit Prioritized Access Protocol for Wireless Sensor Networks. In *RTSS'02*.

[47] T. W. Carley, M. A. Ba, R. Barua, and D. B. Stewart. Contention-free periodic message scheduler medium access control in wireless sensor/actuator networks. In *RTSS '03*.

[48] Dick Caro. *Wireless Networks for Industrial Automation*. ISA Press, 2004.

[49] Amit Chakrabarti, Chandra Chekuri, Anuptam Gupta, and Amit Kumar. Approximation Algorithms for the Unsplittable Flow Problem. In *Approximation Algorithms for Combinatorial Optimization*, volume 2462, pages 51–66. Springer Berlin Heidelberg, 2002.

[50] Jae-Hwan Chang and Leandros Tassiulas. Routing for maximum system lifetime in wireless ad-hoc networks. In *37th Ann. Allerton Conf. Comm., Control, and Computing*, September 1999.

[51] Jae-Hwan Chang and Leandros Tassiulas. Energy conserving routing in wireless ad-hoc networks. In *INFOCOM*, pages 22–31 vol.1, 2000.

[52] Jae-Hwan Chang and Leandros Tassiulas. Maximum lifetime routing in wireless sensor networks. *IEEE/ACM Trans. Netw.*, 12(4):609–619, 2004.

[53] D Chen, M Nixon, and A Mok. *WirelessHART^{TM} Real-Time Mesh Network for Industrial Automation*. Springer, 2010.

[54] Feng Chen, T Talanis, R German, and F Dressler. Real-time enabled ieee 802.15.4 sensor networks in industrial automation. In *SIES '09: IEEE International Symposium on Industrial Embedded Systems*, pages 136–139, Jul 2009.

[55] Lijun Chen, Steven H. Low, and John C. Doyle. Joint congestion control and media access control design for ad hoc wireless networks. In *INFOCOM '05: Proceedings of the 24th IEEE Conference on Computer Communications*, pages 2212–2222, 2005.

[56] Wei Cheng, Xiuzhen Cheng, Taieb Znati, Xicheng Lu, and Zexin Lu. The complexity of channel scheduling in multi-radio multi-channel wireless networks. In *INFOCOM '09: The 28th IEEE Conference on Computer Communications*, pages 1512–1520, Apr 2009.

[57] Krishna Kant Chintalapudi. i-MAC - a MAC that learns. In *IPSN '10*.

[58] O. Chipara, Z. He, Guoliang Xing, Qin Chen, Xiaorui Wang, Chenyang Lu, J. Stankovic, and T. Abdelzaher. Real-time Power-Aware Routing in Sensor Networks. In *14th IEEE International Workshop on Quality of Service (IWQoS'06)*, pages 83–92, June 2006.

[59] Octav Chipara, Chenyang Lu, and Gruia-Catalin Roman. Real-time query scheduling for wireless sensor networks. In *RTSS '07*.

[60] Octav Chipara, Chenyang Lu, and John Stankovic. Dynamic conflict-free query scheduling for wireless sensor networks. In *IEEE International Conference on Network Protocols(ICNP'06)*, pages 321–331, 2006.

[61] Octav Chipara, Chengjie Wu, Chenyang Lu, and William Griswold. Interference-Aware Real-Time Flow Scheduling for Wireless Sensor Networks. In *ECRTS'11*.

[62] H. Choi, J. Wang, and E.A. Hughes. Scheduling on sensor hybrid network. In *ICCCN '05: Proceedings of the 14th International Conference on Computer Communications and Networks*, pages 503–508, Oct. 2005.

[63] Citysense. http://www.citysense.net/.

[64] R. Cogill and H. Hindi. Optimal routing and scheduling in flexible manufacturing systems using integer programming. In *46th IEEE Conference on Decision and Control*, pages 4095–4102, Dec. 2007.

[65] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[66] Crossbow Technology. TelosB mote platform. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf`.

[67] C.M. De Dominicis, P. Ferrari, A. Flammini, E. Sisinni, M. Bertocco, G. Giorgi, C. Narduzzi, and F. Tramarin. Investigating WirelessHART coexistence issues through a specifically designed simulator. In *I2MTC '09: International Instrumentation and Measurement Technology Conference*, Singapore, May 2009.

[68] Sheetalkumar Doshi, Shweta Bhandare, and Timothy X Brown. An on-demand minimum energy routing protocol for a wireless ad hoc network. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(3):50–66, 2002.

[69] Marco F. Duarte and Yu Hen Hu. Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing*, 64(7):826–838, Jul. 2004.

[70] Enrique J. Duarte-melo and Mingyan Liu. Data-gathering wireless sensor networks: Organization and capacity. *Computer Networks*, 43:519–537, 2003.

[71] Jack Edmonds and Richard M. Karp. Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. *Journal of the ACM*, 1972.

[72] Sinem Coleri Ergen and Pravin Varaiya. TDMA scheduling algorithms for wireless sensor networks. *Wireless Networks*, May 2009.

[73] Tullio Facchinetti, Luis Almeida, Giorgio C. Buttazzo, and Carlo Marchini. Real-time resource reservation protocol for wireless mobile ad hoc networks. In *RTSS '04*.

[74] Emad Felemban, Chang-Gun Lee, and Eylem Ekici. MMSPEED: Multipath Multi-SPEED Protocol for QoS Guarantee of Reliability and Timeliness in Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, 5(6):738–754, 2006.

[75] P. Ferrari, A. Flammini, D. Marioli, S. Rinaldi, E. Sisinni, and A. Taroni. An innovative distributed instrument for WirelessHART testing. In *I2MTC '09: International Instrumentation and Measurement Technology Conference*, Singapore, May 2009.

[76] Steven Fortune, John Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111 – 121, 1980.

[77] Robert Fourer, David M. Gay, and Brian W. Kernighan. *AMPL A Modeling Language for Mathematical Programming*. Duxbury Press/Brooks/Cole Publishing Company, 2003.

[78] G. Franchino and G. Buttazzo. WBuST: A real-time energy-aware MAC layer protocol for wireless embedded systems. In *ETFA'12*.

[79] Alvin Fu, Eytan Modiano, and John N. Tsitsiklis. Optimal energy allocation for delay-constrained data transmission over a time-varying channel. In *INFOCOM*, 2003.

[80] Yong Fu, Mo Sha, Chengjie Wu, Andrew Kutta, Chenyang Lu, Humberto Gonzalez, Anna Leavey, Weining Wang, Bill Drake, Yixin Chen, and Pratim Biswas. Thermal Modeling for a HVAC Controlled Real-life Auditorium. In *International Conference on Distributed Computing Systems (ICDCS14)*, July 2014.

[81] Shashidhar Gandham, Milind Dawande, and Ravi Prakash. Link scheduling in sensor networks: distributed edge coloring revisited. In *INFOCOM*, 2005.

[82] Shashidhar Gandham, Ying Zhang, and Qingfeng Huang. Distributed minimal time convergecast scheduling in wireless sensor networks. In *ICDCS '06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, pages 50–50, 2006.

[83] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[84] D. Gay, P. Levis, R. V. Behren, M. Welsh, E. Brewer, and D. Culler. The nesC Language: A Holistic Approach to Network Embedded Systems. In *ACM PLDI*, 2003.

[85] Amitabha Ghosh, Ozlem Durmaz Incel, Anil Kumar, and Bhaskar Krishnamachari. Multi-channel scheduling algorithms for fast aggregated convergecast in sensor networks. In *MASS '09: Proceedings of the 6th IEEE International Conference on Mobile Ad hoc Sensor Systems*, 2009.

[86] Omprakash Gnawali, Ki-Young Jang, Jeongyeup Paek, Marcos Vieira, Ramesh Govindan, Ben Greenstein, August Joki, Deborah Estrin, and Eddie Kohler. The tenet architecture for tiered sensor networks. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 153–166, New York, NY, USA, 2006. ACM.

[87] Joël Goossens, Shelby Funk, and Sanjoy Baruah. Priority-driven scheduling of periodic task systems on multiprocessors. *Real Time Systems*, 25(2-3):187 – 205, 2003.

[88] Yu Gu, Tian He, Mingen Lin, and Jinhui Xu. Spatiotemporal delay control for low-duty-cycle sensor networks. In *RTSS '09*.

[89] Nan Guan, Martin Stigge, Wang Yi, and Ge Yu. New response time bounds for fixed priority multiprocessor scheduling. In *RTSS '09*.

[90] Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-optimal sensor placements in gaussian processes. In *ICML*, 2005.

[91] B. Hajek and G. Sasaki. Link scheduling in polynomial time. *IEEE Transactions on Information Theory*, 34(5):910–917, Sep 1988.

[92] Sohail Hameed and Nitin H. Vaidya. Log-time algorithms for scheduling single and multiple channel data broadcast. In *MobiCom '97: Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking*, pages 90–99, 1997.

[93] Song Han, Jianping Song, Xiuming Zhu, Aloysius K. Mok, Deji Chen, Mark Nixon, Wally Pratt, and Veena Gondhalekar. Wi-HTest: Compliance test suite for diagnosing devices in real-time WirelessHART network. In *RTAS '09: Proceedings of the 15th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 327–336, 2009.

[94] Song Han, Xiuming Zhu, Aloysius K. Mok, Deji Chen, and Mark Nixon. Reliable and Real-time Communication in Industrial Wireless Mesh Networks. In *RTAS'11*.

[95] M G Harbour and J.C. Palencia. Response time analysis for tasks scheduled under edf within fixed priorities. In *RTSS '03*.

[96] David A. Hayes, Michael Rumsewicz, and Lachlan L. H. Andrew. Quality of service driven packet scheduling disciplines for real-time applications: Looking beyond fairness. pages 405–412, 1999.

[97] Tian He, Brian M. Blum, Qing Cao, John A. Stankovic, Sang H. Son, and Tarek F. Abdelzaher. Robust and timely communication over highly dynamic sensor networks. *Real-Time Systems*, 37(3), 2007.

[98] Tian He, John A. Stankovic, Chenyang Lu, and Tarek Abdelzaher. SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks. In *ICDCS '03*.

[99] F. Hu, X. Cao, and C. May. Optimized scheduling for data aggregation in wireless sensor networks. In *ITCC '05: International Conference on Information Technology: Coding and Computing*, volume 2, pages 557–561, April 2005.

[100] S.C.-H. Huang, Peng-Jun Wan, Xiaohua Jia, Hongwei Du, and Weiping Shang. Minimum-latency broadcast scheduling in wireless ad hoc networks. In *INFOCOM '07: 26th IEEE International Conference on Computer Communications*, pages 733–739, 2007.

[101] O. Durmaz Incel, P.G. Jansen, and S.J. Mullender. MC-LMAC: A multi-channel MAC protocol for wireless sensor networks. Technical Report TR-CTIT-08-61, Centre for Telematics and Information Technology, University of Twente, 2008. `http://doc.utwente.nl/65085`.

[102] O.D. Incel and B. Krishnamachari. Enhancing the data collection rate of tree-based aggregation in wireless sensor networks. In *SECON '08: 5th IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 569–577, Jun 2008.

[103] Ozlem Durmaz Incel, Amitabha Ghosh, Bhaskar Krishnamachari, and Krishna Kant Chintalapudi. Multi-channel scheduling for fast convergecast in wireless sensor networks. Technical Report CENG-2008-9, University of Southern California, 2009. `http://anrg.usc.edu/www/publications/papers/CENG-2008-9_TechReport.pdf`.

[104] Praveen Jayachandran and Matthew Andrews. Minimizing End-to-End Delay in Wireless Networks Using a Coordinated EDF Schedule. In *INFOCOM'10*.

[105] Z Jindong, L Zhenjun, and Z Yaopei. ELHFR: a graph routing in industrial wireless mesh network. In *Proceedings of the 2009 IEEE International Conference on Information and Automation*, Zhuhai/Macau, China, Jun 2009.

[106] Ramesh Johari and John N. Tsitsiklis. Efficiency loss in a network resource allocation game. *Journal Mathematics of Operations Research*, 29(3):407–435, 2004.

[107] Petr Jurcík, Ricardo Severino, Anis Koubâa, Mário Alves, and Eduardo Tovar. Real-time communications over cluster-tree sensor networks with mobile sink behaviour. In *RTCSA '08*.

[108] K. Kalpakis, K. Dasgupta, and P. Namjoshi. Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks. *Computer Networks*, 42:697–716, 2003.

[109] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly. Distributed multi-hop scheduling and medium access with delay and throughput constraints. In *MobiCom '01*.

[110] Kyriakos Karenos and Vana Kalogeraki. Real-time traffic management in sensor networks. In *RTSS '06*.

[111] Kyriakos Karenos, Vana Kalogeraki, and Srikanth V. Krishnamurthy. A rate control framework for supporting multiple classes of traffic in sensor networks. In *RTSS '05*.

[112] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *STOC'84*.

[113] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems*. Springer, 2004.

[114] A.N. Kim, F. Hekland, S. Petersen, and P. Doyle. When HART goes wireless: Understanding and implementing the WirelessHART standard. In *ETFA '08: IEEE International Conference on Emerging Technologies and Factory Automation*, pages 899–907, Sep 2008.

[115] Youngmin Kim, Hyojeong Shin, and Hojung Cha. Y-MAC: An energy-efficient multi-channel MAC protocol for dense wireless sensor networks. In *IPSN '08: Proceedings of the 7th international conference on Information processing in sensor networks*, pages 53–63, 2008.

[116] K. Klues, G. Hackmann, O. Chipara, and C. Lu. A component-based architecture for power-efficient media access control in wireless sensor networks. In *IPSN*, 2007.

[117] H.-J. Korber, H. Wattar, and G. Scholl. Modular wireless real-time sensor/actuator network for factory automation applications. *IEEE Transactions on Industrial Informatics*, 3(2):111–119, May 2007.

[118] Anis Koubaa, M Alves, and Eduardo Tovar. i-GAME: An implicit GTS allocation mechanism in ieee 802.15.4 for time-sensitive wireless sensor networks. In *ECRTS '06*.

[119] A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, 134(6):516–526, 2008.

[120] A. Krause, B. McMahan, C. Guestrin, and A. Gupta. Robust submodular observation selection. *JMLR*, 9:2761–2801, Dec 2008.

[121] Lakshman Krishnamurthy, Robert Adler, Phil Buonadonna, Jasmeet Chhabra, Mick Flanigan, Nandakishore Kushalnagar, Lama Nachman, and Mark Yarvis. Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 64–75. ACM, 2005.

[122] Ariel Kulik, H. Shachnai, and Tami Tamir. Maximizing submodular set functions subject to multiple linear constraints. In *SODA*, 2009.

[123] Nai-Luen Lai, Chung-Ta King, and Chun-Han Lin. On maximizing the throughput of convergecast in wireless sensor networks. *Lecture Notes in Computer Science*, 5036:396–408, 2008.

[124] H. Lee, A. Cerpa, and P. Levis. Improving wireless simulation through noise modeling. In *IPSN*, 2007.

[125] Huang Lee and A. Keshavarzian. Towards energy-optimal and reliable data collection via collision-free scheduling in wireless sensor networks. In *INFOCOM '08: 27th IEEE Conference on Computer Communications*, pages 2029–2037, Apr 2008.

[126] S. Lee, B. Bhattacharjee, and S. Banerjee. Effcient geographic routing in multihop wireless networks. In *MobiHoc*, 2005.

[127] Tomas Lennvall, Stefan Svensson, and Fredrik Hekland. A comparison of WirelessHART and ZigBee for industrial applications. In *WFCS '08: Proc. of the 7th IEEE International Workshop on Factory Communication Systems*, 2008.

[128] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *Sensys*, 2003.

[129] B. Li, Z. Sun, K. Mechitov, C. Lu, S. Dyke, G. Agha, and B. Spencer. Realistic case studies of wireless structural control. In *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS'13)*, April 2013.

[130] Bo Li, Lanshun Nie, Chengjie Wu, and Humberto Gonzalez Chenyang Lu. Incorporating Emergency Alarms in Reliable Wireless Process Control. In *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS'15)*, April 2015.

[131] Fei Li. Competitive scheduling of packets with hard deadlines in a finite capacity queue. In *INFOCOM*, 2009.

[132] Huan Li, P. Shenoy, and K. Ramamritham. Scheduling communication in real-time sensor applications. In *RTAS'04*.

[133] Huan Li, P. Shenoy, and K. Ramamritham. Scheduling Messages with Deadlines in Multi-Hop Real-Time Sensor Networks. In *RTAS'05*.

[134] Q. Li and R. Negi. Scheduling in multi-hop wireless networks with priorities. *CoRR*, abs/0901.2922, 2009.

[135] Qun Li, Javed Aslam, and Daniela Rus. Online power-aware routing in wireless ad-hoc networks. In *MobiCom*, 2001.

[136] Xiaojun Lin and S. Rasool. A distributed joint channel-assignment, scheduling and routing algorithm for multi-channel ad-hoc wireless networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications*, pages 1118–1126, May 2007.

[137] Ke Liu, Nael Abu-Ghazaleh, and Kyoung-Don Kang. JiTS: Just-in-time scheduling for real-time sensor data dissemination. In *PERCOM '06*.

[138] Chenyang Lu, Brian M. Blum, Tarek F. Abdelzaher, John A. Stankovic, and Tian He. RAP: A real-time communication architecture for large-scale wireless sensor networks. In *RTAS '02*.

[139] Lin Ma, Kunal Agrawal, and Roger D. Chamberlain. A memory access model for highly-threaded many-core architectures. *Future Generation Computer Systems*, 30:202–215, January 2014.

[140] Lin Ma, Kunal Agrawal, and Roger D. Chamberlain. Analysis of classic algorithms on GPUs. In *Proc. of the 12th ACM/IEEE Int'l Conf. on High Performance Computing and Simulation (HPCS)*, 2014.

[141] Lin Ma and Roger D. Chamberlain. A performance model for memory bandwidth constrained applications on graphics engines. In *Proc. of Int'l Conf. on Application-specific Systems, Architectures and Processors*, 2012.

[142] Rahul Mangharam, Anthony Rowe, Raj Rajkumar, and Ryohei Suzuki. Voice over sensor networks. In *RTSS '06*.

[143] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. The Flooding Time Synchronization Protocol. In *SenSys'04*.

[144] T. Melodia, D. Pompili, and I.F. Akyildiz. Optimal local topology knowledge for energy efficient geographical routing in sensor networks. In *Proc. IEEE INFOCOM'04*, 2004.

[145] K. Menger. Zur allgemeinen kurventheorie. *Fund. Math.*, 10:96 – 115, 1927.

[146] L. Montestruque and M. Lemmon. CSOnet: A metropolitan scale wireless sensor-actuator network. In *MODUS '08: International Workshop on Mobile Device and Urban Sensing*, 2008.

[147] Thomas Moscibroda. The worst-case capacity of wireless sensor networks. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 1–10. ACM, 2007.

[148] Joseph Naor, Adi Rosén, and Gabriel Scalosub. Online time-constrained scheduling in linear networks. In *INFOCOM*, pages 855–865, 2005.

[149] A. Nedic and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48 –61, Jan. 2009.

[150] S. M. Shahriar Nirjon, John A Stankovic, and Kamin Whitehouse. Iaa: Interference aware anticipatory algorithm for scheduling and routing periodic real-time streams in wireless sensor networks. In *Seventh International Conference on Networked Sensing Systems (INSS), 2010*.

[151] R.S. Oliver and G. Fohler. Probabilistic Estimation of End-to-End Path Latency in Wireless Sensor Networks. In *MASS'09*.

[152] Avi Ostfeld and et al. The Battle of the Water Sensor Networks (BWSN): A Design Challenge for Engineers and Algorithms. *Journal of Water Resources Planning and Management*, 134(6):556–568, 2008.

[153] Meng-Shiuan Pan and Yu-Chee Tseng. Quick convergecast in ZigBee beacon-enabled tree-based wireless sensor networks. *Comput. Commun.*, 31(5):999–1011, 2008.

[154] W. Pattara-Atikom, P. Krishnamurthy, and S. Banerjee. Distributed mechanisms for quality of service in wireless lans. *IEEE Wireless Communications*, 2003.

[155] N. Pereira, B. Andersson, E. Tovar, and A. Rowe. Static-priority scheduling over wireless networks with multiple broadcast domains. In *RTSS '07*.

[156] Joonas Pesonen, Haibo Zhang, Pablo Soldati, and Mikael Johansson. Methodology and tools for controller-networking co-design in WirelessHART. In *14th IEEE International Conference on Emerging Technologies and Factory Automation (EFTA)*, Mallorca, Spain, Sep 2009.

[157] C.G. Prohazka. Decoupling link scheduling constraints in multi-hop packet radio networks. *IEEE Transactions on Computers*, 38(3):455–458, Mar 1989.

[158] Michael Rabbat and Robert Nowak. Distributed optimization in sensor networks. In *IPSN*, 2004.

[159] S. Ramanathan. A unified framework and algorithm for (T/F/C)DMA channel assignment in wireless networks. In *INFOCOM '97: 16th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 900–907, Apr 1997.

[160] S. Ramanathan and E. L. Lloyd. On the complexity of link scheduling in multi-hop radio networks. In *Proceedings of the 26th Conference on Information Science and Systems*, Mar 1992.

[161] Subramanian Ramanathan and Errol L. Lloyd. Scheduling algorithms for multihop radio networks. *IEEE/ACM Trans. Netw.*, 1(2):166–177, 1993.

[162] R. Ramaswami and K.K. Parhi. Distributed scheduling of broadcasts in a radio network. In *INFOCOM '89: Proceedings of the Eighth Annual Joint Conference of the IEEE Computer and Communications Societies. Technology: Emerging or Converging*, pages 497–504, Apr 1989.

[163] Lei Rao, Xue Liu, Jian-Jia Chen, and Wenyu Liu. Joint Optimization of System Lifetime and Network Performance for Real-Time Wireless Sensor Networks. In *QSHINE*, pages 317–333, 2009.

[164] Injong Rhee, Ajit Warrier, Jeongki Min, and Lisong Xu. DRAND: distributed randomized TDMA scheduling for wireless ad-hoc networks. In *MobiHoc '06: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pages 190–201. ACM, 2006.

[165] A. Rowe, R. Mangharam, and R. Rajkumar. RT-Link: A time-synchronized link protocol for energy- constrained multi-hop wireless networks. In *SECON '06*.

[166] Abusayeed Saifullah, Paras Tiwari, Mo Sha, Dolvara Gunatilaka, Bo Li, Chengjie Wu, Chenyang Lu, and Yixin Chen. Delay Analysis for Reliable and Real-Time Wireless Sensor-Actuator Networks. Technical Report WUCSE-2014-55, Washington University in St. Louis, 2014.

[167] Abusayeed Saifullah, Chengjie Wu, Paras Tiwari, You Xu, Yong Fu, Chenyang Lu, and Yixin Chen. Near Optimal Rate Selection for Wireless Control Systems. In *RTAS'12*.

[168] Abusayeed Saifullah, Chengjie Wu, Paras Babu Tiwari, You Xu, Yong Fu, Chenyang Lu, and Yixin Chen. Near optimal rate selection for wireless control systems. *ACM Transactions on Embedded Computing Systems (TECS'14)*, April 2014.

[169] Abusayeed Saifullah, You Xu, Yixin Chen, and Chenyang Lu. End-to-End Communication Delay Analysis in Industrial Wireless Networks. *IEEE Transactions on Computers*, 99, May 2014.

[170] Abusayeed Saifullah, You Xu, Chenyang Lu, and Yixin Chen. End-to-end delay analysis for fixed priority scheduling in WirelessHART networks. In *RTAS'11*.

[171] Abusayeed Saifullah, You Xu, Chenyang Lu, and Yixin Chen. Priority Assignment for Real-time Flows in WirelessHART networks. In *ECRTS'11*.

[172] Abusayeed Saifullah, You Xu, Chenyang Lu, and Yixin Chen. Real-Time Scheduling for WirelessHART Networks. In *RTSS'10*.

[173] Jens B. Schmitt and Utz Roedig. Sensor network calculus - A framework for worst case analysis. In *DCOSS '05*.

[174] K. Seada, M. Zuniga, A. Helmy, and B. Krishnamachari. Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks. In *SenSys*, 2004.

[175] M.M. Sekhar and K.N. Sivarajan. Routing and scheduling in packet radio networks. In *IEEE International Conference on Personal Wireless Communications*, pages 335–339, 2000.

[176] Mo Sha, Dolvara Gunatilaka, Chengjie Wu, and Chenyang Lu. Implementation and Experimentation of Industrial Wireless Sensor-Actuator Network Protocols. In *The 12th European Conference on Wireless Sensor Networks (EWSN'15)*, February 2015.

[177] Weiping Shang, Pengjun Wan, and Xiaodong Hu. Approximation algorithm for minimal convergecast time problem in wireless sensor networks. *Wireless Networks*, Sep 2009.

[178] Suresh Singh, Mike Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *MobiCom*, 1998.

[179] Pablo Soldati and Mikael Johansson. A mathematical programming approach to deadline-constrained transmission scheduling in WirelessHART networks. In *Swedish National Control Conference '08*.

[180] Pablo Soldati, Haibo Zhang, and Mikael Johansson. Deadline-constrained transmission scheduling and data evacuation in WirelessHART networks. In *ECC '09*.

[181] J. Song, S. Han, A. K. Mok, D. Chen, M. Lucas, and M. Nixon. A study of process data transmission scheduling in wireless mesh networks. In *ISA EXPO Technical Conference*, October 2007.

[182] J. Song, A. K. Mok, D. Chen, and M. Nixon. Challenges of wireless control in process industry. In *Workshop on Research Directions for Security and Networking in Critical Real-Time and Embedded Systems,*, 2006.

[183] J. Song, A. K. Mok, D. Chen, M. Nixon, T. Blevins, and W.Wojsznis. Improving pid control with unreliable communications. *ISA EXPO Technical Conference*, 2006.

[184] Jianping Song, Song Han, Aloysius K. Mok, Deji Chen, Mike Lucas, and Mark Nixon. WirelessHART: Applying Wireless Technology in Real-Time Industrial Process Control. In *RTAS '08*.

[185] Marco Spuri. Holistic Analysis for Deadline Scheduled Real-Time Distributed Systems. Technical report, 1996.

[186] J.A. Stankovic, T.E. Abdelzaher, Chenyang Lu, Lui Sha, and J.C. Hou. Real-time communication and coordination in embedded sensor networks. *Proceedings of the IEEE*, 91(7):1002–1022, July 2003.

[187] John A. Stankovic, Krithi Ramamritham, and Marco Spuri. *Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms*. Kluwer Academic Publishers, 1998.

[188] I. Stojmenovic. Localized network layer protocols in wireless sensor networks based on optimizing cost over progress ratio. *IEEE Network*, 20(1):21–27, 2006.

[189] Ivan Stojmenovic and Xu Lin. Power-aware localized routing in wireless networks. *IEEE Trans. Parallel Distrib. Syst.*, 12(11):1122–1133, 2001.

[190] Sundar Subramanian, Sanjay Shakkottai, and Ari Arapostathis. Broadcasting in sensor networks: The role of local information. In *INFOCOM*, 2006.

[191] L. Tassiulas and A. Ephremides. Jointly optimal routing and scheduling in packet ratio networks. *IEEE Transactions on Information Theory*, 38(1):165–168, Jan 1992.

[192] E. Toscano and L. Lo Bello. Multichannel Superframe Scheduling for IEEE 802.15.4 Industrial Wireless Sensor Networks. *IEEE Transactions on Industrial Informatics*, 2012.

[193] Hua-Wen Tsai and Tzung-Shi Chen. Minimal time and conflict-free schedule for convergecast in wireless sensor networks. In *ICC '08: IEEE International Conference on Communications*, pages 2808–2812, May 2008.

[194] Yu-Chee Tseng and Meng-Shiuan Pan. Quick convergecast in Zigbee/IEEE 802.15.4 tree-based wireless sensor networks. In *MobiWac '06: Proceedings of the 4th ACM international workshop on Mobility management and wireless access*, pages 60–66, 2006.

[195] Adrian Vetta. Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions. In *FOCS*, 2002.

[196] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25 – 57, 2006.

[197] Xiaodong Wang, Xiaorui Wang, Xing Fu, Guoliang Xing, and Nitish Jha. Flow-based real-time communication in multi-channel wireless sensor networks. In *EWSN '09*.

[198] Andreas Willig. Recent and emerging topics in wireless industrial communications: A selection. *IEEE Trans. on Industrial Informatics*, 2007.

[199] Wirelesshart. http://www.hartcomm.org/.

[200] Chengjie Wu, Mo Sha, Dolvara Gunatilaka, Abusayeed Saifullah, C. Lu, and Y. Chen. Analysis of EDF Scheduling for Wireless Sensor-Actuator Networks. In *IWQoS'14*.

[201] Chengjie Wu, You Xu, Yixin Chen, and Chenyang Lu. Submodular Game for Distributed Application Allocation in Shared Sensor Networks. In *INFOCOM*, 2012.

[202] Chengjie Wu, Ruixi Yuan, and Hongchao Zhou. A novel load balanced and lifetime maximization routing protocol in wireless sensor networks. In *VTC2008-spring*, 2008.

[203] Jie Wu, Shuhui Yang, and M. Cardei. On maintaining sensor-actor connectivity in wireless sensor and actor networks. In *INFOCOM '08: The 27th IEEE Conference on Computer Communications*, pages 286–290, April 2008.

[204] S. Wu and K.S. Candan. Gper: Geographic power efficient routing in sensor networks. In *ICNP*, 2004.

[205] Yafeng Wu, J.A. Stankovic, Tian He, and Shan Lin. Realistic and efficient multi-channel communications in wireless sensor networks. In *INFOCOM '08: the 27th Conference on Computer Communications. IEEE*, pages 1193–1201, Apr 2008.

[206] Lijie Xu, Jiannong Cao, Shan Lin, Haipeng Dai, Xiaobing Wu, and Guihai Chen. Energy-efficient Broadcast Scheduling with Minimum Latency for Low-Duty-Cycle Wireless Sensor Networks. In *MASS'13*.

[207] Yinsheng Xu, Fengyuan Ren, Tao He, Chuang Lin, Canfeng Chen, and Sajal K. Das. Real-time routing in wireless sensor networks: A potential field approach. *ACM Transactions on Sensor Networks*, 9(3):35:1–35:24, June 2013.

[208] You Xu, Abusayeed Saifullah, Yixin Chen, Chenyang Lu, and S. Bhattacharya. Near optimal multi-application allocation in shared sensor networks. In *MobiHoc*, 2010.

[209] Bo Yu, Jianzhong Li, and Yingshu Li. Distributed data aggregation scheduling in wireless sensor networks. In *INFOCOM*, 2009.

[210] Hua Yu, Prasant Mohapatra, and Xin Liu. Channel assignment and link scheduling in multi-radio multi-channel wireless mesh networks. *Mob. Netw. Appl.*, 13(1-2):169–185, 2008.

[211] Haibo Zhang, Fredrik Osterlind, Pablo Soldati, Thiemo Voigt, and Mikael Johansson. Rapid convergecast on commodity hardware: Performance limits and optimal policies. In *SECON '10*.

[212] Haibo Zhang, Fredrik Osterlind, Pablo Soldati, Thiemo Voigt, and Mikael Johansson. Time-optimal convergecast with separated packet copying: Scheduling policies and performance. Technical Report TRITA-EE 2009:050, Automatic Control Lab, Royal Institute of Technology (KTH), Stockholm, Sweden, May 2009.

[213] Haibo Zhang and Hong Shen. Energy-efficient beaconless geographic routing in wireless sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 21(6):881 –896, June 2010.

[214] Haibo Zhang, Pablo Soldati, and Mikael Johansson. Optimal link scheduling and channel assignment for convergecast in linear WirelessHART networks. In *WiOpt'09*.

[215] Haibo Zhang, Pablo Soldati, and Mikael Johanssonn. Efficient link scheduling and channel hopping for convergecast in WirelessHART networks. Technical Report TRITA-EE 2009:018, Automatic Control Lab, Royal Institute of Technology (KTH), Stockholm, Sweden, Jan 2009.

[216] Hongwei Zhang, Anish Arora, Young-ri Choi, and Mohamed G. Gouda. Reliable bursty convergecast in wireless sensor networks. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 266–276, 2005.

[217] Ying Zhang, Shashidhar Gandham, and Qingfeng Huang. Distributed minimal time convergecast scheduling for small or sparse data sources. pages 301–310, 2007.

[218] G. Zhou, C. Huang, T. Yan, T. He, J. A. Stankovic, and T. F. Abdelzaher. MMSN: Multi-frequency media access control for wireless sensor networks. In *INFOCOM '06: 25th IEEE International Conference on Computer Communications*, pages 1–13, Apr 2006.

[219] H. Zhu, M. Li, I. Chlamtac, and B. Prabhakaran. A survey of quality of service in ieee 802.11 networks. *IEEE Wireless Communications*, 2004.

[220] Qi Zhu, Yang Yang, Eelco Scholte, Marco Di Natale, and Alberto Sangiovanni-Vincentelli. Optimizing extensibility in hard real-time distributed systems. In *RTAS '09: Proceedings of the 2009 15th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 275–284, 2009.

[221] X Zhu, W Dong, A K. Mok, S Han, J Song, D Chen, and M Nixon. A location-determination application in WirelessHART. In *International Workshop on Real-Time Computing Systems and Applications*, pages 263–270. IEEE Computer Society, 2009.

[222] Marco Zuniga and Bhaskar Krishnamachari. Analyzing the transitional region in low power wireless links. In *SECON*, 2004.

Vita

Chengjie Wu

Degrees

Ph.D Computer Science, December 2014
M.S. Computer Science, August 2013
M.S. Control Science and Engineering, May 2008
B.S. Math and Physics, May 2006

Publications

Bo Li, Lanshun Nie, Chengjie Wu, Humberto Gonzalez and Chenyang Lu (2015). Incorporating Emergency Alarms in Reliable Wireless Process Control. *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS'15)*, April 2015.

Mo Sha, Dolvara Gunatilaka, Chengjie Wu and Chenyang Lu (2015). Implementation and Experimentation of Industrial Wireless Sensor-Actuator Network Protocols. *European Conference on Wireless Sensor Networks (EWSN'15)*, February 2015.

Yong Fu, Mo Sha, Chengjie Wu, Andrew Kutta, Chenyang Lu, Humberto Gonzalez, Anna Leavey, Weining Wang, Bill Drake, Yixin Chen, and Pratim Biswas (2014). Thermal Modeling for a HVAC Controlled Real-life Auditorium. *IEEE International Conference on Distributed Computing Systems (ICDCS'14)*, July 2014.

Chengjie Wu, Mo Sha, Dolvara Gunatilaka, Abusayeed Saifullah, Chenyang Lu, and Yixin Chen (2014). Analysis of EDF Scheduling for Wireless Sensor-Actuator Networks. *ACM/IEEE International Symposium on Quality of Service (IWQoS'14)*, May 2014.

Abusayeed Saifullah, Chengjie Wu, Paras Tiwari, You Xu, Yong Fu, Chenyang Lu and Yixin Chen (2014). Near Optimal Rate Selection for Wireless Control Systems, *ACM Transactions on Embedded Computing Systems, Special Issue on Real-Time and Embedded Technology and Applications (TECS'14)*, Volume 13, Issue 4s, pp. 128:1-128:25, April 2014.

Abusayeed Saifullah, Chengjie Wu, Paras Tiwari, You Xu, Yong Fu, Chenyang Lu and Yixin Chen (2012). Near Optimal Rate Selection for Wireless Control Systems, *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'12)*, April 2012.

Chengjie Wu, You Xu, Yixin Chen and Chenyang Lu (2012). Submodular Game for Distributed Application Allocation in Shared Sensor Networks, *IEEE International Conference on Computer Communications (INFOCOM'12)*, March 2012.

Octav Chipara, Chengjie Wu, Chenyang Lu and William Griswold (2011). Interference-Aware Real-Time Flow Scheduling for Wireless Sensor Networks, *Euromicro Conference on Real-Time Systems (ECRTS'11)*, July 2011.

December 2014