TECHNISCHE
UNIVERSITÄT
DARMSTADT

IMPROVING QUALITY OF SERVICE IN WIRELESS SENSOR NETWORKS FOR INDUSTRIAL AUTOMATION

Am Fachbereich Informatik
der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Dissertationsschrift

von

DINGWEN YUAN, M.SC.

Geboren am 23. November 1981 in Shanghai, China

Erstreferent: Prof. Dr. -Ing. Matthias Hollick

Korreferent: Dr. -Ing. Utz Roedig

Tag der Einreichung: 10. Dezember 2015
Tag der Disputation: 25. Januar 2016

Darmstadt, 2016
Hochschulkennziffer D17

ABSTRACT

Monitoring and control systems have played a central role in industry and everyday life, often in a non-intrusive manner. Yet, they will become more ubiquitous, autonomous and distributed, with the rapid development of the envisioned technologies of "smart homes", "smart cities" and "industry 4.0". All these new technologies are built upon the enabling technology of Wireless Sensor Networks (WSNs). Their success depends to a large extent on the communication capability of WSNs. Fast reaction and feedback is a common characteristic of these technologies, therefore, how to achieve low-latency, high-reliability and flexibility in WSN communication is a key challenge, and a decisive success factor.

WSN refers to a network that connects a number of low-cost, low-power sensor nodes which have sensing and/or actuating capabilities, and can communicate with each other over short distance via a low-power radio. The predominant advantages that WSN offers are 1) distribution and fault tolerance in communication and sensing/actuation by leveraging large number of sensor nodes, 2) cost reduction by removing the cables of communication and power supply, and 3) flexibility in the deployment of tiny cableless sensor nodes. However, one main drawback of the wireless technology, in contrast to the mature wired counterpart, is the much weaker communication capability — the combined result of stronger interference in the wireless channel, the weak signal strength of low-power radios and the complexity in the scheduling of multi-hop wireless communication.

The main goal of the thesis is to facilitate the transition from wired technology to wireless technology for industrial automation. Specifically, I provide solutions for improving and guaranteeing Quality of Service (QoS) in WSN communication.

I tackle the problem for two scenarios where the network topology is either known or not. When the network topology is known, I adopt an approach of reservation-based scheduling, i.e., through centralized scheduling of communication opportunities, in order to optimize various communication metrics. In the thesis, I propose a very efficient multi-channel scheduling algorithm that gives nearly optimal latency performance (within 1.22% of the optimum) for the tree-based convergecast, which is by far the predominant communication pattern, especially for monitoring applications. I also propose very efficient multi-channel scheduling algorithms that offer high schedulability and low overhead for multi-flow periodic real-time communication on an arbitrary network topology with multiple gateways. Such a communication pattern is typical of a multi-loop control system.

On the other hand, if the network topology is unknown or changes very dynamically, I optimize the QoS in communication by exploiting concurrent transmission on the physical layer, which is routing-free by nature. First, I proposes a simple model for concurrent transmissions in WSN which accurately predicts the success or failure in the packet reception. Then I design the Sparkle protocol for highly reliable, low latency and energy efficient multi-flow periodic communication. Finally, it presents the Ripple protocol for high throughput, reliable and energy efficient network flooding using

pipeline transmissions and forward error correction, which significantly improves the state-of-the-art.

Although the thesis assumes WSN as the communication technology and industrial automation as the application scenario, it is by no means restricted to these settings since the proposed solutions can be applied to other wireless networks and other scenarios with similar communication patterns and QoS concerns.

ZUSAMMENFASSUNG

Überwachungs- und Steuerungssysteme sind von zentraler Bedeutung sowohl in der Industrie als auch im alltäglichen Leben. Oftmals arbeiten sie in kaumwahrnehmbarer Art und Weise. Dennoch werden sie mit der rasanten Entwicklung der aufkommenden Technologien "Smart Homes", "Smart Cities" und "Industrie 4.0" noch allgegenwärtiger, autonomer und verteilter. All diese neuen Technologien basieren auf drahtlosen Sensornetzen (WSNs). Ihr Erfolg hängt zu einem großen Teil von der Kommunikationsfähigkeit von drahtlosen Sensornetzen ab. Schnelle Reaktion und Rückkopplung sind eine gemeinsame Charakteristik dieser Technologien, weshalb das Erreichen einer geringen Latenz, einer hohen Zuverlässigkeit sowie Flexibilität bei der Kommunikation in drahtlosen Sensornetzen eine große Herausforderung und ein entscheidender Erfolgsfaktor ist.

Ein drahtloses Sensornetz repräsentiert ein Netzwerk bestehend aus einer Reihe von kostengünstigen Sensorknoten mit wenig Leistung, welche über Mess- und/oder Aktuatorfähigkeiten verfügen. Diese Knoten kommunizieren drahtlos miteinander über kurze Distanzen mittels eines leistungsschwachen Funkmoduls. Die Hauptvorteile, die ein drahtloses Sensornetz bietet sind 1) verteilte und fehler-tolerante Kommunikation und Messungen/Regelungen aufgrund der großen Anzahl von Sensorknoten, 2) Kostenreduzierung durch Entfernen der Verkabelungen für Kommunikation und Energieversorgung und 3) Flexibilität bei der Anbringung von winzigen drahtlosen Sensorknoten. Ein Hauptnachteil der drahtlosen Technologie im Gegensatz zum ausgereiften drahtgebundenen Gegenstück ist jedoch die viel schwächere Kommunikationsfähigkeit — der kombinierte Effekt von stärkerer Interferenz beim drahtlosen Medium, die schwache Signalstärke der leistungsschwachen Funkmodule und die Komplexität beim Koordinieren der drahtlosen Kommunikation mit mehreren Zwischenempfängern.

Das Hauptziel dieser Dissertation ist es, die Transition von der drahtgebundenen zur drahtlosen Technologie im Rahmen der industriellen Automatisierung zu erleichtern. Im Einzelnen stelle ich Lösungen bereit, um die Dienstgüte (QoS) bei der Kommunikation in drahtlosen Sensornetzen zu garantieren und zu verbessern.

Ich löse das Problem für zwei Szenarien, bei denen die Netzwerktopologie entweder bekannt oder unbekannt ist. Im ersteren Fall wende ich einen Ansatz basierend auf festgelegter Koordinierung an, d.h. die Kommunikation wird mittels eines zentralisierten Algorithmus bestimmt, um verschiedene Kommunikationsmetriken zu optimieren. In dieser Dissertation stelle ich einen sehr effizienten, mehrkanaligen Koordinierungsalgorithmus vor, der nahezu optimale Latenzperformance (innerhalb von 1,22% des Optimums) für den baumbasierten Sammelempfang ermöglicht, welcher bei weitem das am häufigsten verwendete Kommunikationsmuster darstellt, besonders für Überwachungsapplikationen. Darüber hinaus präsentiere ich sehr effiziente, mehrkanalige Koordinierungsalgorithmen, welche hohe Koordinierungsfähigkeit und geringen Overhead für periodische Echtzeitkommunikation mittels mehreren Datenflüssen in einer beliebigen Netztopologie mit mehreren Gateways bieten. Ein solches Kommunikationsmuster ist typisch für ein Steuerungssystem mit mehreren Schleifen.

Andernfalls, wenn die Netzwerktopologie nicht bekannt ist oder sich sehr dynamisch ändert, optimiere ich die Kommunikationsdienstgüte durch Ausnutzen der simultanen Übertragung auf der physikalischen Ebene, die von Natur aus kein Routing erfordert. Zuerst schlägt ich ein einfaches Modell für simultane Übertragungen in drahtlosen Sensornetzen vor, welches den Erfolg oder Nicht-Erfolg des Zustellens eines Paketes zuverlässig vorhersagt. Danach entwickle ich das "Sparkle"-Protokoll für höchst zuverlässige, energie-effiziente periodische Kommunikation mit mehreren Datenflüssen bei geringer Latenz. Abschließend präsentiere ich das "Ripple"-Protokoll für hohen Datendurchsatz sowie zuverlässiges und energie-effizientes Netzwerk-Flooding mittels Pipeline-Übertragungen und Vorwärtsfehlerkorrektur, welches den bisherigen Stand der Technik signifikant verbessert.

Obwohl die Dissertation von drahtlosen Sensornetzen als Kommunikationstechnologie und industrieller Automatisierung als Anwendungsszenario ausgeht, sind die vorgestellten Lösungen nicht auf diese Annahmen beschränkt. Sie können auf andere drahtlose Netze und andere Szenarien mit ähnlichen Kommunikationsmustern und Dienstgüteanforderungen übertragen werden.

ACKNOWLEDGMENTS

First of all, I would like to thank Prof. Matthias Hollick for his trust in my capability in doing research and his constant motivation and advice during my Ph.D. study. Without his effort, this thesis would not be possible. I would also like to thank Dr. Utz Roedig for his willingness to be my co-advisor and his comments and suggestions to this thesis. In addition, I want to express my gratitude to the rest of the thesis committee — Prof. Alejandro Buchmann, Prof. Gerhard Neumann and Prof. Oskar von Stryk.

Moreover, I would also like to thank all the colleagues in the *Secure Mobile Networking Lab*. The technical discussion and happy social time spent together are unforgettable. Of course, the uncountable cakes you made has sweetened my working time. Especially, I would like to thank Dr. Michael Riecker for being an accommodating office mate and a good friend.

I want to thank my wife Xiaomin and our daughter Wanru for tolerating my sometime absent-mindedness. Many thanks also to my friends in different places on the world. Last but not least, I would like to thank my family back in Shanghai for their understanding, encouragement and support.

Darmstadt, 2015 Dingwen

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

PREVIOUSLY PUBLISHED MATERIAL

This thesis includes material previously published in peer-reviewed publications. In accordance with the regulations of the Computer Science department at TU Darmstadt, I list below the chapters which include verbatim fragments from these publications.

Chapter 4

- Section 4.1 revises "Tree-based Multi-channel Convergecast in Wireless Sensor Networks" by Dingwen Yuan and Matthias Hollick. In *Proceedings of the 13th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2012.
- Section 4.2 revises and extends "Optimization and Scheduling of Wireless Sensor Networks for Periodic Control Systems" by Dingwen Yuan and Matthias Hollick. In *Proceedings of the 11th GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze"*, 2012.

Chapter 5

- Section 5.1 revises "Let's Talk Together: Understanding Concurrent Transmission in Wireless Sensor Networks" by Dingwen Yuan and Matthias Hollick. In *Proceedings of the 38th IEEE Conference on Local Computer Networks (LCN)*, 2013.
- Section 5.2 revises "Making 'Glossy' Networks Sparkle: Exploiting Concurrent Transmissions for Energy Efficient, Reliable, Ultra-low Latency Communication in Wireless Control Networks" by Dingwen Yuan, Michael Riecker and Matthias Hollick. In *Proceedings of the 11th European Conference on Wireless Sensor Networks (EWSN)*, 2014.
- Section 5.3 revises "Ripple: High-throughput, Reliable and Energy-efficient Network Flooding in Wireless Sensor Networks" by Dingwen Yuan and Matthias Hollick. In *Proceedings of the 16th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2015.

1 INTRODUCTION

1.1 MOTIVATION

Wireless Sensor Networks (WSNs) are a technology that is undergoing rapid development. It was born as the result of the vision of ubiquitous computing, the rapid progress in miniaturization of Integrated Circuits (IC), Micro-electro-mechanical Systems (MEMS) and low power wireless communication technology [ASSC02]. A WSN is typically composed of a number of small devices, called sensor nodes, which are small low-power computer systems with constrained computation power and memory, often equipped with various sensors and actuators. The combination of the capabilities of sensing/actuating, computation and communication makes WSN a versatile general-purpose system for numerous applications.

WSNs have been successfully applied to a large amount of interesting applications, such as environmental and habitat monitoring, smart houses, medical assistance, traffic monitoring, and battlefield surveillance, as well as industrial automation, to name a few. Industrial automation is in general a very challenging scenario for WSNs because both high reliability and stringent deadline requirements of packet delivery need to be satisfied simultaneously. In addition, some applications of industrial automation may have further requirements on high energy efficiency or large throughput. If the communication requirements cannot be satisfied, the control systems will not perform as expected, which will cause failure in production, and in the extreme case even cause injury and casualty to humans. This explains the reason why the industrial automation society is nowadays still largely relying on wired technology such as Fieldbus and real-time Ethernet [Tho05, MT07], and the penetration of WSN technology is limited to applications, of which the Quality of Service (QoS) requirements are relatively relaxed.

The WSN technology provides many advantages in comparison with its wired counterpart. The most obvious and direct one is that it totally or to a large extent saves the wiring cost. Though it is argued in [ÅGB11] that quite some actuators in process automation today are pneumatic and these actuators must be powered by the main grid since the wireless sensor nodes are unable to provide sufficient energy for them, it is typically much easier to wire power than to wire communication. The removal of wiring saves not only large investment, but also much space and weight, which is especially precious in locations such as aircrafts [YB11]. Second, after the removal of cabling, the flexibility in the deployment of sensor nodes greatly improves, especially at places where it is hard or even impossible to do wiring, e.g., on the rotating parts of manufacturing robots. As a result, many secondary process variables that have long been unmeasurable can now be easily measured. Also temporal measurements that are experimental or diagnostic can now be easily carried out [ÅGB11]. Both new possibilities of measurement will contribute to the quality of industrial automation. Third, quite often the network structure of WSNs is ad-hoc and the large number of sensor nodes provide fault tolerance to a system. This will facilitate a more and

more distributed architecture in industrial automation. Finally, WSNs can be more energy efficient than the wired counterpart due to the broadcasting nature of wireless communication. The small energy consumption may even be totally covered by energy harvesting, especially because the sensor nodes are quite often deployed near working places where large quantity of energy in different forms such as vibration, pressure and high temperature, is available [CC08]. The highly energy efficient technology of WSN is more sustainable and beneficial to the environment.

In fact, the wiring saving and the distribution of system architecture are two continuous evolutions from the initiation of industrial automation. The first automation systems were based on mainframe computers, star-like connected with devices via miles of cables, but it only allowed for polling communication initiated by the central computer [SW07]. Today's standard technology of Fieldbus or Ethernet based automation systems use fewer cables and allow devices to communicate with each other in a more distributed fashion. I can predict that in the future automation systems may be totally cableless, and the communication will be more distributed as the devices become more autonomous, and can perform local control based on the ambient information. These highly autonomous systems are indispensable for the realization of the ideas of "smart homes", "smart cities" and "industry 4.0".

However, the flexibility in deployment, high level of distribution, low power consumption and low production cost of WSNs comes at the price of weak communication capability. This is arguably the biggest obstacle to the wide adoption of WSNs in industrial automation. The de facto low-power radio standard IEEE 802.15.4 [Soco6] operates mostly in the freely available and therefore strongly interfered 2.4 GHz Industrial, Scientific and Medical (ISM) band, which exacerbates the interference problem. For instance, in the same frequency band, the transmission power of the almost ubiquitous IEEE 802.11 (WiFi) can be hundreds of times higher than that of IEEE 802.15.4 [MET08]. Although there are technologies available and upcoming, e.g., Ultra-wideband (UWB) communication, which are not limited to the ISM band, they are immature or not in wide use. A large amount of wireless devices work only in the 2.4 GHz ISM band and we have to live with the technology for a long time. Therefore, it is important to design protocols that interconnect these devices. Apart from that, the throughput, latency and reliability of the low-power wireless communication are far worse than those of the wired technology. Let alone, for relatively large wireless networks, multi-hop communication is unavoidable. The performance degrades due to the increase of hops and the mutual interference among sensor nodes. Hence, how to guarantee a sufficient communication QoS for industrial automation in a very noisy radio frequency band with low-power radios becomes a very challenging problem.

1.2 COMMUNICATION REQUIREMENTS OF INDUSTRIAL AUTOMATION

For all industrial automation systems except the most trivial, communication is indispensable. This is due to the following facts: 1) the sensor data of monitoring systems need to be delivered to the control center, which is typically not co-located with the sensors; 2) the controllers, sensors and actuators of control systems can not be positioned at the same place, but sensors and actuators have to be in physical proximity to the process they are interfacing with. Currently, nearly all industrial monitoring and control systems are *digital systems*. This means that the signals (messages) between

the controllers, sensors and actuators are quantized and the systems are largely discrete-time systems, i.e., the communication and the activation of controllers occur periodically. Besides, event-based communication may happen aperiodically. In this thesis, I will only focus on digital systems.

1.2.1 *Communication Delay and Period*

The performance of monitoring and control systems depends to a large degree on the system delay. For monitoring systems, it refers to the time span between the sensing of the related information and the delivery of it to the display units. For control systems, it refers to the time span between the sensing of the related information and the corresponding actuation. Although the system delay also depends on other parameters such as the execution time of the control algorithm, it can never be smaller than the communication delay. In this thesis, I only focus on the typical digital systems with sampling periods that are larger than or equal to the system delay, i.e., these systems have implicit or restricted deadline according to the real-time computing jargon. In case that the sampling period is smaller than the system delay, one can also analyze the system performance and stability. However, techniques such as pipeline transmission need to be applied since the new round of communication is activated before the old one finishes.

Generally, for both monitoring and control systems, a shorter sampling period (or a higher sampling rate) typically improves the system performance at the cost of more resource consumption, such as computation, communication and energy. Therefore, how to pick a suitable sampling period is a trade-off.

Since there is no stability problem, the sampling period of monitoring systems can be solely determined by the user requirement. For example, in a chemical reaction system, the liquid level alarms typically allow lower delay than the measurements of the slowly varying temperature. On the other hand, for control systems, overly large periods not only deteriorate the performance, but may also cause system instability. According to the Nyquist sampling theorem, for a signal band-limited to f_B, the sampling rate should be at least $2f_B$. This rule is not very useful for control systems, since 1) the system model is normally approximations; 2) the system has noise and disturbance and 3) the input signals such as steps are not band-limited. There exist a few best practices for the selection of sampling periods [SS11]. One example is that the sampling period T should be selected in the range of

$$\frac{1}{40f_c} < T < \frac{1}{10f_c}, \tag{1.1}$$

where f_c is the closed-loop bandwidth, i.e., the frequency at which the closed-loop gain has decreased by 3dB, or is about 70% of the DC gain. Specifically, I propose two general approaches for industrial wireless communication in this thesis. The approach of centralized scheduling has at least 10 milliseconds communication delay if the industrial standard of WirelessHART [HAR07] is applied, which has a slot length of 10 milliseconds. Thus, it is suitable for control systems with closed-loop bandwidth lower than 10 Hz. The approach of concurrent transmission pushes the minimum communication delay to about 1 millisecond. Accordingly, it is suitable for control systems with closed-loop bandwidth lower than 100 Hz.

There is a large amount of industrial control systems suitable for the wireless communication proposed in this thesis. Almost all process control systems fall in this category, because the control variables have rather slow dynamics and typically require no shorter than 1 second of sampling period. For example, the flow rate, which is a fast control variable in process control, needs a sampling period of 1 to 3 seconds while the temperature only needs a sampling period of 10 to 180 seconds [LZ06]. On the contrary, the proposed wireless communication is unsuitable for motion control applications that control the position, speed and acceleration of mechanical systems with electric motors. These control systems are so-called isochronous real-time systems and have typical sampling period of 250 µs to 1 ms [PN09].

The communication delay of a control system is composed of two parts: the sensor-to-controller delay and the controller-to-actuator delay. If the sensors and the controller are time synchronized and the sensor data is time-stamped, the sensor-to-controller delay can be compensated by an estimator that reconstructs the undelayed plant states [ZBP01]. Hence, it is preferable to place the controller in proximity to the actuators to reduce the controller-to-actuator delay.

1.2.2 *Communication Reliability*

In addition to the communication delay requirement, the communication reliability requirement is also key to the performance of monitoring and control systems. Similar to the effect of communication delay, a higher communication reliability improves the system performance, while a lower communication reliability may cause control system instability.

The concrete requirement of reliability depends on the type of the application and the type of the message. For example, an emergency alarm signifying a variable over the set limit should be transmitted reliably to the controller in a short deadline so that the emergency reaction can be reliably triggered, for safety operation guarantees. It is much more dangerous to miss an emergency alarm than a periodic sensor reading.

Traditional wired solutions such as Fieldbus provide perfect reliability, hence communication reliability is not an issue at all. However, shifting to wireless communications, it is impossible to have perfect reliability while giving delay guarantee at the same time. Therefore, the so-called communication and control co-design is of increasing importance. Recent research demonstrates that by applying advanced control algorithms, the communication reliability requirement of control systems can be greatly reduced. For example, Li et al. [LMW+16] observe that a control design combining an observer based on the extended Kalman filter and the model prediction control can effectively compensate the communication loss. A case study based on the simulation of an exothermic chemical reaction plant shows that the control system performs satisfactorily even with 60% packet loss on both of the sensor-to-controller and controller-to-actuator communication. Moreover, the loss of the actuation commands has larger impact on the system performance than the loss of the sensor data. Additionally, there is a trade-off between the sampling period and the communication reliability requirement for control systems. A control system can work stably under a lower reliability requirement by shortening the sampling period [ZBP01]. Generally, the solutions proposed in this thesis should offer high enough communication reliability for most industrial automation applications.

1.3 GOALS

The communication pattern of industrial automation, whether it is used for monitoring and supervision, or for controlling, is typically periodic. The focus of the thesis is to optimize and to guarantee QoS in periodic multi-hop WSN communication, especially to achieve high reliability, high throughput and high energy-efficiency in communication while satisfying the requirements on hard deadline, thereby helping to remove the main obstacle in the application of WSNs to industrial automation and bringing us closer to the destination of wide adoption of WSNs in industrial automation. The general goals of the thesis are to:

1. Propose novel algorithms and protocols for the combined goals of performance optimization and hard deadline guarantee in industrial WSNs.
2. Verify and evaluate the effectiveness of the proposed algorithms and protocols through simulation and real-world experiments.
3. Push forward the performance in comparison to the state-of-the-art solutions.

1.4 CONTRIBUTIONS

The contributions of the thesis are mainly in optimizing performance while guaranteeing hard deadlines of periodic multi-hop communication in WSNs, which is essential for the wide adoption of WSNs in industrial automation.

Specifically, the contributions can be divided into two parts: 1) improving QoS in WSN communication with centralized scheduling, and 2) improving QoS in WSN communication by exploiting concurrent transmissions.

1.4.1 *Improving Communication QoS with Centralized Scheduling*

WirelessHART [HAR07] is the most popular industrial standard of wireless sensor networks for industrial automation. The key feature of WirelessHART is that it is a Time Division Multiple Access (TDMA) protocol which allows the simultaneous use of multiple orthogonal channels, but disallows spatial reuse of the same channel. The spatial reuse of the same channel refers to using the same channel simultaneously on more than one link that are spatially separated enough such that the mutual interference cannot destroy any of the transmissions. The feature of disabling spatial reuse is essential to ensuring high reliability in communication — a must for industrial automation, because on the one hand, the dynamic change of wireless channel condition makes the guarantee of spatial separation a heavyweight process and on the other hand the mutual interference among concurrently scheduled links needs to be minimized. In addition, disabling spatial reuse simplifies the WirelessHART scheduling.

WirelessHART specifies the basic communication paradigm, but leaves open the concrete scheduling algorithms for the TDMA protocol [CNM10]. Thus, how to design a suitable TDMA scheduling is the first step in employing WirelessHART. This thesis proposes highly effective and lightweight scheduling algorithms for two categories of communications that are widely used in industrial automation: 1) tree-based

periodic data collection and data distribution [1], and 2) mesh-based periodic multi-flow communication where a flow refers to an end-to-end pair. Typically, the industrial monitoring and supervision applications assume the first communication pattern while the industrial control applications assume the second communication pattern. Nevertheless, the first communication pattern may also be used for industrial control applications.

My main contributions to the tree-based periodic data collection and data distribution [YH12b] are two fold:

1. I derive an Integer Programming (IP)-based optimal solution to the *minimum length scheduling* by taking into account the resource limitations of the maximum buffer size of each node and the total number of channels available. The minimum length scheduling uses the minimum number of TDMA slots for a cycle of communication, thus it also achieves the maximum throughput and minimum latency.

2. Since the optimal solution relies on IP, which is NP-hard, it is only feasible in very small networks. For bigger networks, we have to rely on heuristics. I present a flexible framework for the convergecast/distribution scheduling problem. Based on that, I propose the novel *busy-sender-first* heuristic which is significantly better than the state-of-the-art heuristic in both schedule length and memory consumption as well as being conceptually much simpler.

My main contributions to the problem of periodic mesh-based multi-flow communication with hard deadlines [YH12a] are as follows:

1. I investigate a general system model — arbitrary topology, arbitrary number of gateways, multi-path routing, and end-to-end flows with arbitrary period and hard deadline requirements. Based on this system model, I propose a general framework for scheduling algorithms.

2. I adapt algorithms from the area of multi-processor real-time scheduling to my problem, and identify the best scheduling heuristic Least Laxity First (LLF) with high schedulability rate, low execution time and low memory overhead at intermediate nodes during network operation.

3. I design a simple yet effective *opportunistic aggregation* scheme that works seamlessly with any scheduling algorithm. It substantially increase the schedulability rate.

4. To minimize the overhead of the schedule table, I propose a scheme called *repetitive scheduling* which also works seamlessly with any scheduling algorithm. It incurs negligible schedulability penalty for the case of implicit deadline (i.e., when the period and the deadline of a flow are equal, which is typically the case), but results in very small and scalable size of the schedule table and execution time of a scheduling.

1 I differentiate between the term data distribution and data dissemination, where data distribution refers to the case that a central node sends *different* data to other nodes in the network, and data dissemination refers to the case that a central node sends *same* data to other nodes in the network. Therefore, data dissemination can be viewed as a special case of data distribution

1.4.2 *Improving Communication QoS with Concurrent Transmission*

The physical layer technology of concurrent transmission that appeared recently was shown to be a promising mechanism for low latency and high reliability communication in WSNs. It was first demonstrated by the seminal network flooding protocol, Glossy [FZTS11] to be able to achieve high reliability, low latency and accurate time synchronization simultaneously.

In this thesis, I first investigate the underlying principles of concurrent transmission in WSNs and propose a simple model for it that can accurately predict the reception/loss of concurrent transmissions [YH13]. Then I design the Sparkle protocol [YRH14], which extends the concurrent transmission mechanism, which is originally designed for one-to-all broadcast, to the application of periodic multi-flow communication. The periodic multi-flow communication is end-to-end and is the typical communication pattern of industrial control. Furthermore, Sparkle can control the performance of each communication flow independently with the help of a novel feedback control mechanism. Finally, I propose the Ripple protocol [YH15], which improves the network flooding protocols Glossy and Splash [DCL13a] significantly in terms of throughput and energy efficiency by combining the pipeline transmissions on multiple channels and the efficient Reed-Solomon erasure code.

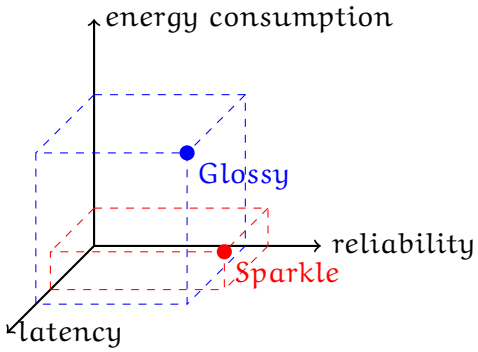


Figure 1.1: QoS comparison between Glossy and Sparkle for end-to-end communication. The QoS metrics of Sparkle are better than those of Glossy, respectively.

My main contributions in this part of the thesis are as follows:

1. I propose an accurate prediction model for the reception/loss of concurrent transmissions, which provides a valuable tool for protocol design and simulation of concurrent transmission in WSNs.

2. I propose Sparkle, a WSN control network design based on concurrent transmission, tailored to periodic multi-loop control systems. It optimizes each end-to-end communication flow by performing "feedback control" on the flow. In Sparkle, I introduce *WSNShape*, a unique topology control technique, that is able to find a reliable end-to-end stripe (instead of a not so reliable single path) based only on the capture effect. I demonstrate through testbed evaluation that Sparkle satisfies the high reliability requirement while in average saves 80% of energy consumption and improves on the latency for 5%, compared to Glossy. The QoS comparison between Sparkle and Glossy is illustrated in Fig. 1.1.

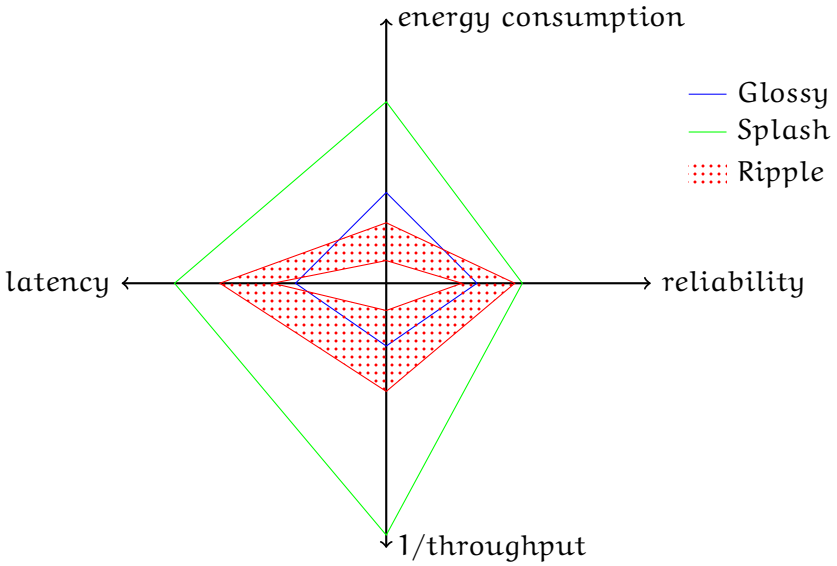


Figure 1.2: QoS comparison among Glossy, Splash and Ripple for flooding communication. A QoS working point corresponds to four line segments closing in a loop. Ripple has a working region while the other two protocols have a single working point. Except the axis *reliability*, the values on the other axes are better when they are smaller.

3. I propose Ripple, which extends the Glossy network flooding with pipeline transmission on multiple channels and forward error correction. I demonstrate through testbed evaluation that Ripple significantly improves on the throughput over 80Kbits/s, a threefold increase compared to the best-case of Glossy. At the same time, Ripple also increases the energy efficiency by a factor of three, compared to Glossy. By tuning the transmission interval, Ripple balances between high reliability and high throughput, suiting a large spectrum of QoS requirements. Finally, applying Reed-Solomon erasure code to Ripple pushes the reliability over that of Glossy, very near to 100%, at the cost of reduced throughput over plain Ripple, but the throughput still doubles, or even triples that of the state-of-the-art data dissemination protocol Splash. The working regions of Ripple, Glossy and Splash, in terms of QoS metrics are illustrated in Fig. 1.2.

1.5 OUTLINE

The thesis contains in total 6 chapters. The structure is outlined in Fig. 1.3 and is structured as follows.

Chapter 1 "*Introduction*" motivates the thesis by describing the QoS problem in applying WSN technology to industrial automation. Then it discusses the main goals and contributions of the thesis.

Chapter 2 "*Related Work*" summarizes the related work in three categories: 1) the status of WSNs for industrial automation, 2) the scheduling for tree and mesh-based wireless sensor networks. These two network topologies are typical and general enough for industrial automation, and 3) the related work on the physical layer technique of concurrent transmission and its effect in improving QoS in communication.

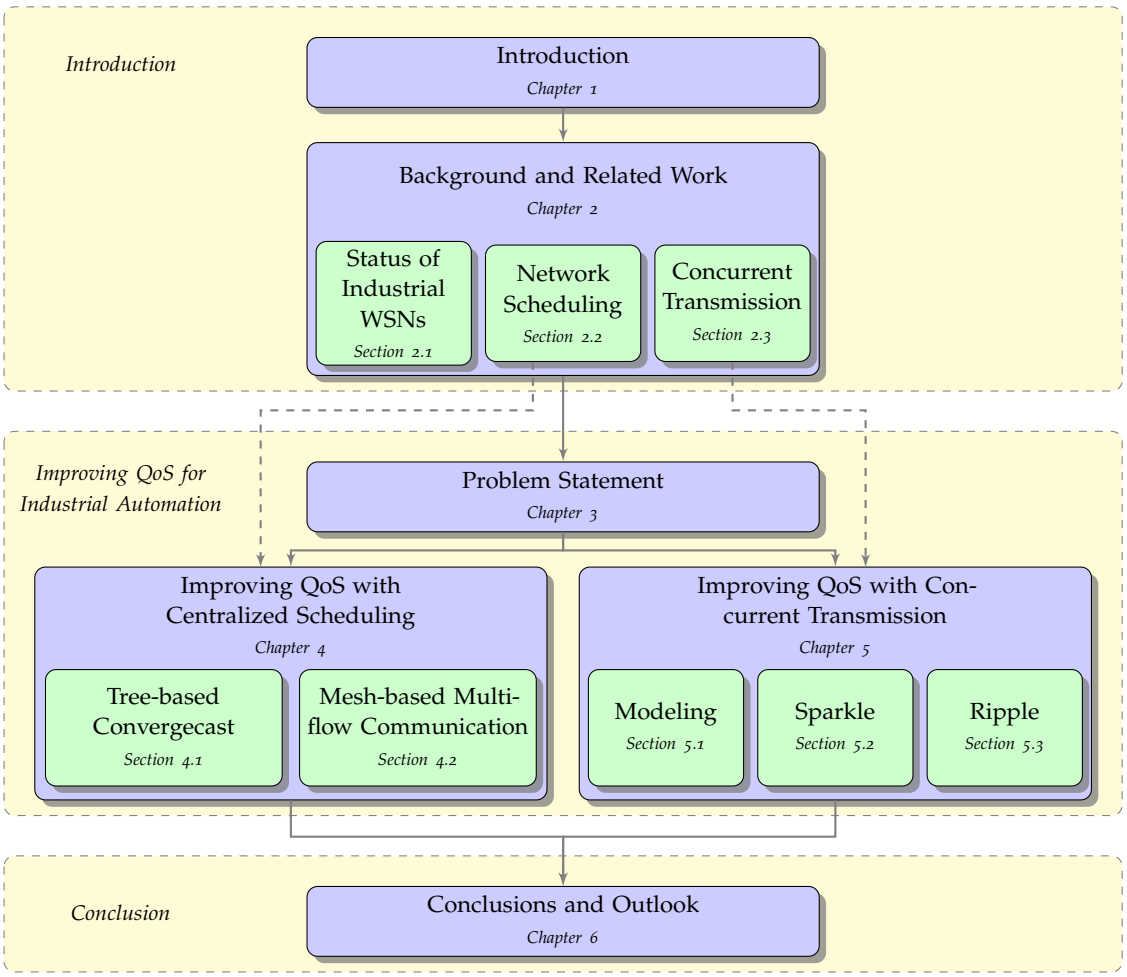


Figure 1.3: Outline of the thesis.

Chapter 3 "*Problem Statement*" proposes a number of key problems that I identify in the related work and that I am going to tackle in this thesis.

Chapter 4 "*Realtime Scheduling of WirelessHART-based Sensor Networks*" deals with the TDMA scheduling for the WirelessHART protocol, which is the de facto wireless standard for industrial automation.

Chapter 5 "*Improving QoS for Industrial Automation with Concurrent Transmission*" deals with the application of the promising physical layer technique of concurrent transmission to WSNs. It is shown through testbed evaluation that with concurrent transmission I can achieve significant improvement in QoS compared to the state-of-the-art of Glossy, such as high throughput, high reliability, low latency and high energy efficiency which is essential for industrial automation. I propose the periodic multi-loop control network Sparkle with flow-based QoS control as well as the high throughput, high reliability and high energy-efficiency network broadcast protocol Ripple.

Chapter 6 "*Conclusions and Outlook*" concludes the thesis and identifies a number of interesting open problems about the industrial WSNs for future research.

2 BACKGROUND AND RELATED WORK

I survey the related work of this thesis from three aspects. First, I investigate the current status of WSN for industrial automation. Here, I look at a selected number of interesting applications, as well as two prominent industrial standards. They serve as the motivation and represent the state-of-the-art of the deployment of WSN in industrial automation, from which I can identify the drawbacks of existing solutions and ways to rectify them. The thesis focus is on two research areas: the centralized TDMA scheduling and the physical layer technique of concurrent transmission. Hence, the second and the third aspects of the related work survey each of them respectively.

2.1 THE CURRENT STATUS OF WIRELESS SENSOR NETWORKS FOR INDUSTRIAL AUTOMATION

Wireless sensor networks have been increasingly applied to industrial automation in recent years. In the following, I survey a number of important works on practical WSN deployments for industrial automation and two prominent industrial standards. These works and standards reflect the current status of wireless sensor networks for industrial automation.

2.1.1 *Practical WSN Technologies for Industrial Automation*

The WSN deployments for industrial automation can be generally divided into two categories: 1) communication with deadline guarantees, and 2) best effort communication.

2.1.1.1 *WSN Deployments with Deadline Guarantees*

The number of works falling in this category is very limited. But deadline guarantee in communication is necessary for industrial automation applications with stringent latency constraints, such as process control and robot control. Otherwise the control system may not be able to satisfy the required performance and may even become instable. This will results in unsatisfactory end-products and in the extreme case, even cause human injury or casualty. Deadline guaranteed communication in WSN is a main focus of this thesis. All protocols proposed in the thesis belong to this category.

A representative work is the GINSENG project [OBB+13, SBR10]. It proposed the GinMAC protocol [SBR10], a tree-based TDMA protocol for industrial automation, which was originally designed for process automation in oil refineries. The protocol guarantees timely delivery of the sensing and actuation data while optimizing reliability and energy consumption. It is suitable for single-rate monitoring and control applications.

GinMAC assumes a fixed tree topology with one sink at the root, where each non-root node of the tree may be occupied by a sensor, an actuator, a combined

sensor and actuator, or a pure relay. Some degree of flexibility in scheduling can be provided by assuming a tree bigger than the actual topology. The TDMA scheduling is static, and is determined offline. The schedule is divided into two parts: the upstream slots, used for the communication from the sensors to the sink, and the downstream slots, used for the communication from the sink to the actuators. The upstream and downstream slots are of three types: the basic slots contain the minimum number of slots for the upstream and downstream traffic; the additional slots are used to improve the reliability, i.e., when transmissions in the basic slots fail, the additional slots can compensate for that; moreover, the unused slots save energy, because in these slots radios are turned off.

GinMAC has limited flexibility in scheduling because it needs to guarantee the hard deadline of data delivery. It uses only one channel, thus under-uses the resource of multiple physical channels provided by the IEEE 802.15.4 standard [Soco6], which could be used to improve throughput and latency. The tree topology does not provide route redundancy, contrary to the mesh topology, although the paper allows *spatial transmission diversity*, i.e., forming multiple trees with the same set of nodes. Finally, GinMAC is not fully adaptive to node join and leave.

2.1.1.2 *WSN Deployments with Best-effort Communication*

There are quite a number of WSN deployments with best-effort communication. These protocols may not be suitable for real-time monitoring and control applications because they provide no hard deadline guarantees. However, they are sufficient for other applications that have relaxed delay requirements. Some representative works are as follows.

The Flush protocol has been designed for the structural health monitoring of the Golden Gate Bridge [KPC+07, KFD+07]. It supports the multi-hop source to sink (end-to-end) efficient bulk data transfer with perfect reliability. With the deployment on the Golden Gate Bridge, it proves effective even for a very long 48-hop line topology. Flush uses pipeline packet transmission with the unique feature of hop-by-hop in-network rate control. The rate control requires that each node dynamically identifies the interfering nodes and determines the transmission rate based on parameters reported by its successor node. The rate control algorithm follows two rules: 1) a node should only transmit when its successor is free from interference, and 2) a node's sending rate cannot exceed the sending rate of its successors. However, Flush only deals with line topology as only one flow is active at a time. Finally, how to perform routing is not covered in the work.

RACNet is a sensor network for monitoring a data center's environmental condition [LLL+09]. It is a very dense WSN with multiple gateways, where each sensor node needs to periodically deliver sensor data to a gateway. The underlying communication protocol is called Wireless Reliable Acquisition Protocol (WRAP), a best effort data collection protocol which combines centralized and distributed mechanisms to achieve scalability and responsiveness. The topology control of WRAP is initiated by the gateways, each running on a different channel for load balancing purpose. Each node listens to the coordinated beaconing broadcasted level by level from the gateways and then selects the best parent. Finally a tree is formed at each gateway where all links are bidirectional. WRAP also implements a distributed algorithm to balance between different trees. The load balancing is performed at the tree with the largest total hop

count — a node in the largest tree will probabilistically leave the tree and joins a tree smaller than average. For data collection, WRAP implements a token-based congestion avoidance mechanism. Specifically, a token is passed in the depth-first order from the root of a tree (a gateway node). A node holding the token is allowed to transmit and packets get aggregated at intermediate nodes. Furthermore, WRAP controls the data streaming rate with similar mechanism to that of Flush [KFD+07]. Finally, to improve end-to-end reliability, WRAP implements a Negative Acknowledgement (NACK)-based, end-to-end data recovery scheme, initiated by the gateways. Evaluation results show that as the aggregate amount of traffic grows, WRAP achieves higher data yields than the open-loop Collection Tree Protocol (CTP) [GFJ+09] and higher total throughput than the Rate-Controlled Reliable Transport Protocol (RCRT) [PG07].

Ceriotti et al. [CCD+11] proposed a WSN application of adaptive lighting control for road tunnels. Only the data collection part of the whole control process is implemented in WSN, i.e., collecting the luminance levels at different positions all along a tunnel. Based on the collected data, a control algorithm sets the lighting intensity of lamps properly in order to meet the legislated curve of the luminance. The downstream data, i.e., the lighting intensity commands are transmitted in a wired network from the Programmable Logic Controller (PLC) to the equipments. Two types of traffic are supported by the WSN: the data collection and the data dissemination. The data collection uses multiple sinks, where each sink periodically and independently builds a collection tree. Each node chooses a parent with the smaller node-to-sink routing cost. High data reliability is achieved with a hop-by-hop recovery scheme. The dissemination protocol allows one-to-many communication from the sinks, by employing a Trickle-like scheme [LPCS04]. It is used to change the light sampling frequency or to modify MAC parameters. Although the data collection is best-effort, it satisfies the communication requirements of the adaptive lighting control with the relaxed control period of 30 seconds. For a control system with shorter control period such as process control, the protocol may be insufficient.

@scale [DLT+12] presented a year-long deployment of 455 wireless plug-load electric meters in a large commercial building. The communication requirement is basically reliable data collection. Specifically, each wireless electric meter needs to deliver a measurement packet every 20 seconds to the datacenter located in the open Internet Protocol version 6 (IPv6) Internet. The peculiarity of @scale is its pure IPv6-based three tier architecture, including 1) the metering tier that is made up of the wireless electric meters, 2) the back haul tier that is made up of multiple Load Balancing Routers (LBRs), and 3) the datacenter tier which runs as a hosted web application. Each wireless electric meter uses an IPv6/6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) stack to connect to an LBR, from which the User Datagram Protocol (UDP) data packets travel over an IPv6 tunnel from the building to the Internet. Finally they reach the database located in the datacenter. Instead of using communication protocols tailored to a WSN application, @scale uses the Internet Protocol (IP) architecture. As a result, the LBRs are transparent to the other tiers which allows the users to deploy new meters and LBRs to increase both the network and backhaul capacity. Furthermore, the LBRs and all meters form a single IPv6 subnet where efficient any-to-any IPv6 routing over constrained links is possible. But the data collection period is of tens of seconds, much longer compared to the protocols proposed in this thesis. Additionally, how the wireless LBRs do load balance and interference avoidance is not covered.

2.1.2 *The Industrial WSN Standards*

The industry shows great interest in applying WSN to industrial automation. I am aware of two WSN standards for this purpose: the WirelessHART [HAR07] and the ISA100.11a [ISA08] standards. The WirelessHART standard was first released by the HART Communication Foundation (HCF) in the HART Field Communication Protocol Specification, Revision 7.0 [HCF07]. The ISA100.11a was ratified as an International Society of Automation (ISA) standard in 2009 [ISA09].

Both standards have a lot in common. The physical layers of both use the IEEE 802.15.4 standard in the ISM 2.4 GHz frequency band [Soco6]. The Media Access Control (MAC) layers are both based on TDMA (Time Division Multiple Access) protocol and channel hopping. To diminish the internal interference, both standards disallow spatial reuse of the same channel at the same time. The TDMA protocol enables the guarantee of latency. The channel hopping improves the robustness of the system since it avoids operating for long time on severely interfered channels. For the routing layer, both use the fault-tolerant mesh topology. A centralized network manager is in charge of the slot scheduling in both standards.

Nevertheless, the WirelessHART and ISA100.11a standards have a number of differences, which are listed in the following.

1. WirelessHART devices are nodes that must have the routing capability while ISA100.11a devices can be pure sensors and actuators without routing capability [PC11, NRR12].

2. The network layer of WirelessHART mainly deals with routing in mesh networks. Similar functionalities are performed by the data link layer in ISA100.11a, while its network layer makes use of the Internet Engineering Task Force (IETF) IPv6 and 6LoWPAN.

3. WirelessHART specifies the slot time to be 10 ms and the ISA100.11a supports a configurable slot time.

4. WirelessHART uses the HART AL for its application layer and ISA100.11a has a more flexible and abstract application layer.

Due to the earlier release time of WirelessHART and the relative simplicity and concreteness, currently WirelessHART has far more standard-compliant product manufacturers than ISA100.11a.

However, in both standards, the functionality of the network manager, i.e., how to perform slot scheduling and channel hopping is not specified in detail. This thesis investigates these problems for a tree convergecast setting and a very general mesh multi-flow communication setting.

2.2 NETWORK SCHEDULING

The network layer of WirelessHART has specified two types of routing: the *graph routing* and the *source routing* [CNM10].

- **Graph Route**. A *graph route* is a subset of the directed links and devices that provides redundant communication routes between a source and a destination

device. The actual route taken is based on current network condition when the packet is conveyed from the source to the destination.

- ▷ **Source Route**. A *source route* is a single directed route (devices and links) between a source and a destination device. The source route is statically specified in the packet itself.

To perform graph routing, a node needs to store the next-hop routing information for a graph ID if it is on the routing graph. When a node gets a packet, it retrieves the graph ID and then routes the packet to the next-hop node. On the other hand, for source routing, all routing information is contained in the data packets. No routing information needs to be stored at intermediate nodes. Both routing mechanisms are powerful but abstract. WirelessHART has specified the "interface" but not the "implementation", i.e., the detailed scheduling algorithms that produce the routing are missing in the standard. The thesis focuses on the graph routing since it causes less overhead for relatively static communication requirement which is mostly the case for industrial automation. Specifically, the thesis provides effective algorithms for the TDMA scheduling of tree-based convergecast and mesh-based multi-flow communication. The former communication pattern is typical for monitoring systems while the latter is typical for control systems.

There are plenty of works on TDMA scheduling for wireless sensor networks with concerns on communication latency. I highlight two categories that are related to industrial automation: 1) *minimum length scheduling*, the goal of which is to achieve minimum number of slots for a round of communication (e.g., each end node needs to transmit a packet to the gateway in a round), and 2) *deadline guaranteed scheduling*, the goal of which is to guarantee that all deadline requirements on communication are satisfied.

2.2.1 *Minimum Length Scheduling*

Convergecast is the most common communication pattern in wireless sensor networks, where a number of source nodes want to send packets to the gateway (sink) node. The minimum length scheduling for convergecast [GZH08, EV10, RCBG10, ZÖS+10, ZSJ09a] uses the minimum number of slots for a round of communication. It is also the scheduling that gives the maximum sampling rate and it normally provides high throughput. Therefore, it is ideal for monitoring applications which rely on convergecast and in which features such as maximum sampling rate and high throughput are desired.

The works of Gandham et al. [GZH08] and Ergen et al. [EV10] investigate the problem of single-channel convergecast scheduling. The first work assumes that the interference range of a node is the same as the communication range. It proposes a distributed scheduling algorithm for tree networks where each node generates exactly one packet. The schedule length of the proposed algorithm is at most $max(3n_k - 1, N)$ slots and the lower bound of schedule length is proved to be $max(3n_k - 3, N)$, where n_k is the maximum number of nodes in any subtree and N is the total number of nodes in the network. Furthermore, it provides a distributed scheduling algorithm for general networks which requires at most 3N slots and in average 1.5N slots. In contrast, the second work assumes a general interference model represented by an

interference graph. It first proves that the problem of convergecast scheduling for tree networks with an arbitrary conflict graph is NP-complete. Then, it proposes two centralized scheduling algorithms drawn from the coloring problem: the node-based scheduling and the level-based scheduling. The evaluation results show that the level-based scheduling works better for topologies with higher density of packets farther away from the sink while the node-based scheduling works better for topologies with equal packet density across the network or higher packet density at low levels.

The works of [RCBG10, ZÖS+10, ZSJ09a] investigate the problem of multi-channel convergecast scheduling. The Packets in Pipe (PIP) protocol by Raman et al. [RCBG10] achieves the minimum schedule length for multi-hop line networks with one source and one sink. The number of channels required is determined by the mutual interference between the nodes. Normally four to five are sufficient. The work by Zhang et al. [ZÖS+10] achieves very high throughput by applying the optimal scheduling and separating the packet copying between the microcontroller and the radio transceiver from the packet transmission. It provides an optimal scheduling algorithm for tree networks where each node generates exactly one packet in each period and there are enough orthogonal channels. The optimal schedule length has taken into account the slots for two-way packet copying and is equal to $max(3n_k - \Delta, N)$ where where n_k is the maximum number of nodes in any subtree and N is the total number of nodes in the network. $\Delta = 1$ if the largest two subtrees have the same number of nodes and $\Delta = 2$ otherwise. Another work of Zhang et al. [ZSJ09a] offers the state-of-the-art solution for multi-channel scheduling of tree networks. It proves that the minimum schedule length is equal to $max(2n_k - 1, N)$ and offers a polynomial time algorithm for minimum length scheduling when given D channels where D is equal to the tree depth. It also presents the state-of-the-art scheduling heuristic for the case that the channel count is less than D. The heuristic produces near-optimal schedule length, but is relatively complex in concept.

2.2.2 *Deadline Guaranteed Scheduling*

Network control systems normally have hard-deadline constraints for communication, therefore the research on *deadline guaranteed scheduling* is a must for these systems.

One of the earliest works on the guarantee of communication deadline in wireless sensor networks is by Caccamo et al. [CZSB02]. It proposes a protocol for scheduling hard-deadline periodic messages and soft-deadline aperiodic messages. It builds on a cellular structure of network where neighboring cells use different frequencies to avoid mutual interference. Nodes within a cell are fully connected, and they run the Earliest Deadline First (EDF) algorithm for implicit medium access of the intra-cell messages. The inter-cell messages are ordered by earliest deadline at router nodes and are scheduled periodically in fixed frames exclusive for directional inter-cell communication. For better performance, a scheme is designed to let the aperiodic messages reuse the reserved frames which are left unused by the hard-deadline messages. The simulation shows that these designs of real-time scheduling have significantly improved the performance of system throughput and latencies of aperiodic messages.

Another example of real-time scheduling in wireless sensor networks is the so-called real-time query scheduling by Chipara et. al [CLR07]. It investigates single gateway networks with arbitrary communication and interference graphs. Single channel based

spatial reuse is used for communication. A query is actually a periodic tree-based convergecast with aggregation. Each query has a deadline and a priority. Two queries are conflict-free if their schedules are separated by at least a certain number of slots. The simulation results show that the slack stealing query scheduling algorithm is effective by combining the benefits of high throughput and deadline guarantees.

2.2.3 *General System Model for Network Scheduling*

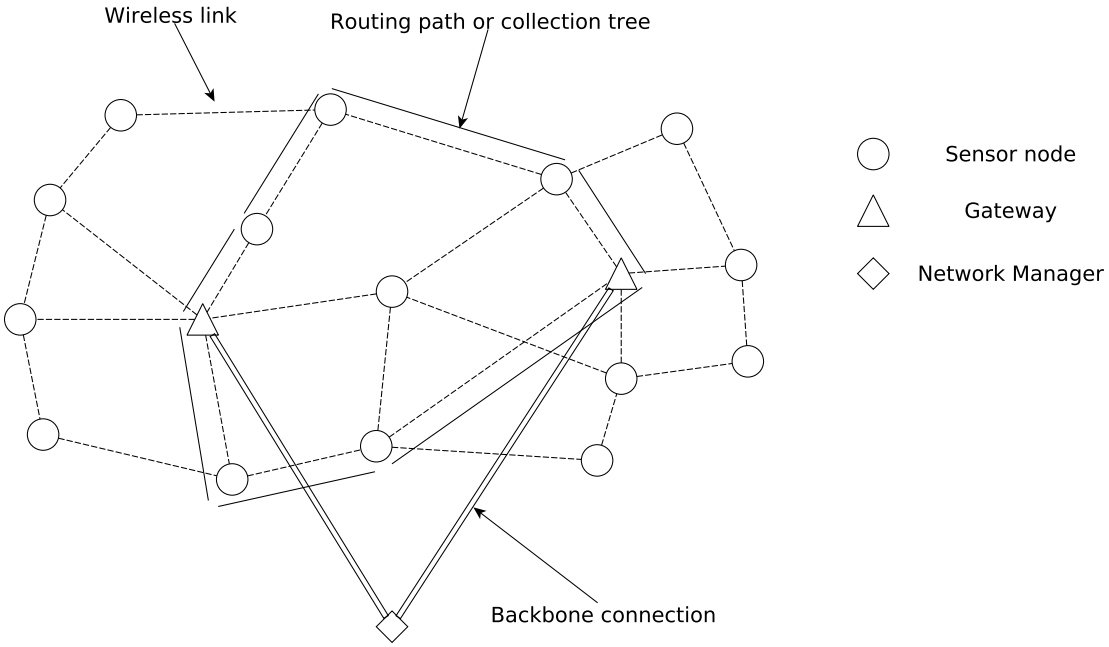


Figure 2.1: General system model for network scheduling.

The general system model for the centralized network scheduling in this thesis is shown in Fig. 2.1. I have formulated an abstract yet powerful system model. The WSN is composed of a number of static/mobile sensor nodes and one or more gateways (sinks). The gateways are connected through a backbone network, which is usually a reliable, high-throughput and low-latency wired network, to a network manager. The network manager is the single central unit, responsible for routing and scheduling. After it has computed the scheduling table, the network manager disseminates it to the whole network through the gateways. At runtime, the sensor nodes periodically deliver the packets to the gateways according to the scheduling, if the communication pattern is convergecast. If the communication pattern is flow-based (end-to-end), the end nodes serving as sensors deliver the sensing packets periodically to the gateways, which function as controllers. Then the gateways will deliver the command packets periodically to the end nodes serving as actuators, according to the scheduling.

2.3 CONCURRENT TRANSMISSIONS IN WIRELESS SENSOR NETWORKS

Wireless sensor networks for industrial automation is a field under active research. Some industrial standards such as WirelessHART [HAR07] and ISA100 [ISA08] have

appeared. A few academic projects have also been reported, such as the GINSENG project [OBB+13], and the TRITon project [CCD+11]. All of them use static or semi-static routing for communication, which unavoidably incurs routing overhead, cannot promptly adapt to the quick change of wireless channel and is inherently not robust to node and link failures.

The recently proposed network flooding protocol Glossy by Ferrari et al. [FZTS11] demonstrates that concurrent transmission in WSNs can achieve the hard and quite often contradictory goals of high reliability, low latency, low energy consumption and accurate time synchronization all at the same time for multi-hop one-to-all communication. It is also robust to node and link failures due to the node diversity. These features match the communication requirements of industrial automation very well. Therefore, this thesis also focuses on exploiting the technique of concurrent transmission to improve the communication QoS in WSN.

2.3.1 *Related Work based on Concurrent Transmissions in WSN*

A few interesting protocol designs are based on concurrent transmission in WSN. They either improve on the original Glossy protocol in terms of communication quality or propose novel services which use Glossy as the communication primitive.

After the Glossy protocol, Ferrari et al. subsequently proposed the Low-power Wireless Bus (LWB) [FZMT12], a protocol that supports one-to-many, many-to-one and many-to-many communication by exploiting the concurrent transmission of Glossy. LWB serves as the wireless counterpart of the Fieldbus which is normally used to connect PLCs (Programmable Logic Controllers), sensors and actuators for device level communication in a plant. The goal of LWB is to achieve communication reliability (dependability) and latency guarantees. The main feature of LWB is that the communication schedule is flooded to all nodes by a centralized host and the communication requests are flooded to the host through contention. In comparison with a number of existing protocols, LWB offers higher throughput, despite that it uses less power.

Doddavenkatappa et al. proposed Splash [DCL13a], a protocol for fast data dissemination in WSN. Instead of the topology-unaware flooding as done in Glossy, it uses a CTP-like [GFJ+09] collection protocol to build the tree structure for flooding. Then, it incorporates a number of techniques — transmission density diversity, opportunistic overhearing, channel cycling, XOR coding and local recovery for higher throughput and reliability in data dissemination. Through testbed evaluation, Splash is shown to be over an order of magnitude faster in data dissemination than the state-of-the-art data dissemination protocols such as DelugeT2 [HC04].

Doddavenkatappa et al. proposed another protocol P^3 [DC14], which is used for reliable end-to-end bulk data transfer. Similar to Splash [DCL13a], P^3 makes use of pipeline transmission, channel cycling and constructive interference. The unique feature is that it can achieve a continuous end-to-end packet transmission, obtaining an average throughput of 178.5 Kbits/s, approaching the data rate of 250Kbits/s of the IEEE 802.15.4 standard. To prepare for the continuous packet transmisison, P^3 needs to collect all link states at the gateway node. With a global view of the network topology, it identifies a number of node-disjoint paths from a source to a destination

node. Then the paths are divided into two areas, and the source node alternately transmits packets into either of them.

A common drawback of Splash and P^3 is the heavyweight process of collecting the global view of network topology at the gateway and disseminating the node-based channel assignment. Additionally, these overheads are not counted in the final calculation of the performance metrics. For example, in a network of 100 nodes, it takes more than 100 seconds to measuring and collecting the link quality measurements as reported in [DC14]. The time is simply too long to assume that the topology will keep stable in the duration. These overheads are avoided in the one-to-all data broadcast protocol Ripple, proposed in this thesis, which uses the novel packet-based channel assignment.

2.4 SUMMARY

In this chapter, I have surveyed the current status of WSN for industrial automation, from both perspectives of deployments and industrial standards. Then, I surveyed the state-of-the-art TDMA scheduling and the state-of-the-art protocols based on concurrent transmission, which greatly improve the communication QoS in WSN.

Based on the related work, I will identify a number of key problems in the next chapter. This thesis tries to tackle these problems. In addition, the related work also provides the state-of-the-art solutions, with which my solutions should compare.

3 PROBLEM STATEMENT

After surveying the related work in the last chapter, I have identified two general methods for improving the communication QoS of WSN for industrial automation. First, I can perform more effective and more lightweight TDMA scheduling to improve the realtime capability of industrial WSNs. Second, I can apply the physical layer technique of concurrent transmission to improve various performance metrics of industrial WSNs.

3.1 HOW TO IMPROVE COMMUNICATION QOS WITH CENTRALIZED SCHEDULING

As stated in the last chapter, WirelessHART is a widely adopted industrial standard for WSNs. Its concurrent use of multiple channels improves the system throughput and its disallowance of spatial reuse of the same channel leads to minimization of internal interference and reduced overhead in tracking network state. WirelessHART does specify the functionality of centralized TDMA scheduling but leaves the detailed algorithm open. Chapter 4 of the thesis will focus on the TDMA scheduling for the WirelessHART protocol.

Monitoring applications are an important category of industrial automation deployments. Quite often, a tree topology is first formed which connects all the sensors to the gateway node at the tree root, before the data collection is performed. A natural question is how to perform the scheduling so that the minimum length scheduling is obtained or approximated. The state-of-the-art solution of the so-called tree convergecast for the WirelessHART protocol is proposed by Zhang et al. [ZSJ09a]. However, though it produces near-optimal schedule length, it has a high complexity. A question to ask is:

- ▷ Is there a heuristic that performs better than the state-of-the-art one by Zhang et al. [ZSJ09a] in terms of shorter schedule length and less memory consumption at intermediate nodes?

The problem will be investigated in Sec. 4.1 and the overall answer is yes.

Besides monitoring, the other important category of industrial automation deployments are the control applications. The communication pattern featuring such applications is periodic multi-flow communication and the goal is to satisfy the deadline requirement for each communication flow. I assume nothing about the network topology, i.e., the network has the most general mesh topology. A specialized form of the WirelssHART scheduling was investigated in the work by Saifullah et al. [SXLC10]. It assumes that there is one gateway in the network and the controller is placed at the gateway. Furthermore, a number of flows (each control loop is mapped to a flow) are to be scheduled where each flow corresponds to the transmission from a sensor node to the gateway and then from the gateway to an actuator. Each flow is activated periodically with potentially different period and has a hard deadline. The problem is

shown to be NP-hard in [SXLC10] and it proposed an effective heuristic Conflict-aware Least Laxity First (CLLF) which gives the best schedulability.

In this thesis, I investigate a more general problem setting — I allow for multiple gateways and multi-path routing. Based on this problem setting, I will investigate the following problems:

- What is the most effective scheduling heuristic in terms of schedulability with short execution time and less memory consumption at intermediate nodes?
- Does the CLLF heuristic still perform well in the aforementioned mesh setting?
- Control packets are generally very short. How will packet aggregation help in improving the schedulability?
- Since the schedule table needs to be transmitted to nodes in the network and to be stored by them, it is imperative to minimize its size. What is an effective way to minimize the schedule table size?

In Sec. 4.2, I will investigate these problems and propose effective algorithms and heuristics for them.

3.2 HOW TO IMPROVE COMMUNICATION QOS WITH CONCURRENT TRANSMISSION

As discussed in the last chapter, concurrent transmission can be exploited to achieve high packet reliability, low communication latency, low energy consumption and accurate time synchronization simultaneously in WSNs. These features match the general communication requirements of industrial automation very well. Therefore, Chapter 5 of the thesis will focus on improving communication QoS with the physical layer technique of concurrent transmission.

Practical implementations of concurrent transmission exist, yet their performance is not well understood due to the lack of expressive models that accurately predict the outcome of packet reception. This poses an obstacle to better utilizing concurrent transmission. Thus, the modeling problem of concurrent transmission that I address in Sec. 5.1 is:

- On which factors does the reception of concurrent transmission depend? Can I create a simple model that can accurately predict the outcome of concurrent transmissions?

The original protocol Glossy which shows the efficacy of concurrent transmission is designed for one-to-all network-wide flooding. The communication paradigm of one-to-all communication does not match the common communication paradigms of industrial automation, which include quite often all-to-one communication for monitoring and one-to-one communication for control. All-to-one communication can be realized with multiple one-to-one communication, therefore, I will focus on one-to-one communication based on concurrent transmission in the thesis.

A naive implementation of one-to-one communication based on concurrent transmission can be done by performing a network-wide Glossy flooding initiated by the source node as done in the LWB protocol [FZMT12]. However, such network-wide

flooding involves all nodes in the network, which results in unnecessarily high energy consumption. Another way to tackle the problem is to measure the network topology as done in the protocol P^3 [DC14], then select only the "necessary" nodes for one-to-one communication. The topology measurement needs to be done continuously, which causes large overhead, but may still be unable to cope with the quick change of the wireless channel and network state. Furthermore, different control loops may have different communication requirements and these requirements may also change at runtime. Always over-provision may not be ideal. Thus, I need to perform "feedback control" on the quality of one-to-one communication. In Sec. 5.2, I will explore the following problems:

- How to perform energy-efficient and reliable one-to-one communication based on concurrent transmission while being very adaptive to the wireless channel and network state change?
- How to perform "feedback control" on the QoS of one-to-one communication?

The one-to-all communication of Glossy successfully improves reliability, latency and energy efficiency in communication simultaneously. Moreover, it has the potential to improve throughput, since only one packet is flooded in each Glossy round. If I would flood more frequently in each round, I could significantly improve the throughput of the system. This is demonstrated by the Splash protocol [DCL13a], which makes use of pipeline transmission on multiple channels. However, Splash relies on the CTP protocol to measure the network topology, based on which it assigns channels statically to each node. Again, this heavyweight process needs to be performed continuously, but may still be unable to cope with the quick change of the network state. Furthermore, I find occasionally that the Glossy may provide unsatisfactory reliability (with packet reception rate $< 90\%$). A natural way to improve the reliability is to use source coding. In summary, the following two problems will be investigated in Sec. 5.3:

- Pipeline transmission on multiple channels greatly improves the throughput of Glossy. How can I perform a lightweight channel assignment which has negligible penalty on the throughput?
- How can the reliability of Glossy be guaranteed/maximized without sacrificing its low latency advantage?

3.3 SUMMARY

This chapter discusses the deficiencies in existing works and the possibilities for improvement. I have identified a number of key problems that I am going to investigate. I provide solutions to the problems in the following chapters. The problems are tackled with two methodologies. The first methodology is about improving communication QoS with centralized TDMA scheduling, which is the main topic of Chapter 4. The second is about improving communication QoS with concurrent transmission, on which Chapter 5 focuses.

4

IMPROVING QOS OF INDUSTRIAL WSN WITH CENTRALIZED SCHEDULING

This chapter deals with the slot scheduling of TDMA-based protocols for wireless sensor networks. The goal of the scheduling is to optimize or to guarantee quality of service (QoS) in industrial WSNs. The basic assumption of the TDMA scheduling is that it uses multiple orthogonal channels but disallows spatial reuse. This is the assumption adopted by WirelessHART [HAR07], the goal of which is to fully utilize the precious channel resource while to minimize the interference that may be brought by the spatial reuse.

The evaluation of the scheduling is based on simulation. The reason is that I want to only focus on the quality of the scheduling, which is well defined in my case. Therefore a simulation based evaluation assures that it is independent of the behaviors of the other parts of the system. Two pieces of work are presented in this chapter: a) tree-based multi-channel convergecast scheduling [YH12b] and b) TDMA scheduling for multi-loop periodic control systems of arbitrary topology [YH12a]. The first work can be applied for single-rate monitoring and control systems with one gateway. Although I only deal with convergecast and convergecast corresponds to the upstream communication (the sensing process) in a control system, the scheduling for the downstream communication (the actuation process) can be easily obtained by reversing the schedule. The second work can be applied to multi-rate monitoring and control systems with one or more gateways. The scheduling goals of both works are different: the first one aims at finding the minimum length scheduling while the second one aims at meeting the hard deadline requirements.

4.1 TREE-BASED MULTI-CHANNEL CONVERGECAST SCHEDULING

4.1.1 *Introduction*

The technological progress in Wireless Sensor Networks has enabled their transition from research environments to industrial application domains such as monitoring and control [SBR10]. These applications are characterized by reliable exchange of periodic data with strict QoS requirements, such as short delay. Standards such as WirelessHART [HAR07], designed specifically for industrial automation, echo the trend.

WirelessHART organizes the sensor nodes into tree or mesh topology and schedules the transmissions using TDMA; the schedule is computed by a centralized network manager and uses multiple orthogonal channels to enable simultaneous transmissions. In order to minimize the interference caused by concurrently scheduled links, WirelessHART abandons spatial reuse and allows only one link to be active on each channel in each timeslot. While it provides a scheduling framework, the actual implementation of scheduling algorithms is left open.

In general, the objectives of minimizing latency and maximizing throughput and reliability requires that scheduling algorithms should be adapted to the given network topology and application domain. Here, I assume a tree-based routing, which is very common for industrial automation because almost all such applications need a centralized controller that handles the control logic. A typical example is the so-called *single-rate control system* [FPW98]. It performs *sensing*, *controlling* and *actuation* with a fixed system rate. In this thesis, I focus on the resulting Tree Convergecast Scheduling with Multiple Channels (TCMC) problem.

Definition 1 (Tree Convergecast Scheduling with Multiple Channels)**.** A part or all of the sensor nodes in a WSN of tree topology generate one packet at the beginning of each period. The packets need to be delivered to the Gateway (GW) using a given number of orthogonal channels. A valid scheduling is the scheduling of a superframe, at the beginning of which all nodes reporting to the GW have one packet in the buffer, and at the end of which all packets have arrived at the GW. Furthermore, the superframe is composed of a number of timeslots, in each of which there is at most one link active on each channel.

In a *single-rate control system*, not only the *sensing*, but also the *actuation* phase can be scheduled with TCMC since the latter is actually an reverse procedure of convergecast. Regarding communication latency, data throughput and system sample rate, *minimum length scheduling* achieves the optimum for all three metrics, as it minimizes the schedule length measured as number of timeslots. By taking into account the resource consumption of WSN, I propose two other scheduling goals — *minimum length and channel scheduling* and *minimum length and buffer size scheduling*. The former refers to the scheduling that, given a fixed buffer constraint, attains the minimum scheduling length while using the smallest possible number of channels. The latter refers to the case that, given a fixed number of channels, attains the minimum scheduling length while requiring smallest worst-case buffer size at sensor nodes. Note, I focus on the problem of minimum scheduling length without taking the packet loss into consideration. The packet reliability and its interplay with the schedule length are left as future works.

A number of solutions to the TCMC problem have been proposed in the literature. Yet, the existing solutions perform either far from the optimal, and/or are very complex to implement. My contributions are as follows:

1. I derive an Integer Programming (IP)-based optimal solution to the *minimum length and buffer size scheduling* and the *minimum length and channel scheduling*. (Sec. 4.1.3)

2. I formulate TCMC scheduling as a decision problem, then create a *general scheduling framework* that is flexible and requires minimal code modification for implementing different scheduling strategies. (Sec. 4.1.4)

3. I propose and implement four heuristics within my framework. I demonstrate that my novel *busy-sender-first* heuristic not only performs significantly better than the state-of-the-art heuristic in both schedule length (the busy-sender-first heuristic is within 1.22% of the optimum) and memory consumption, but also is conceptually much simpler and easier to implement. (Sec. 4.1.4 and Sec. 4.1.5)

4. Based on my results, I derive *guidelines on the optimal configuration* of number of channels and topology of the routing tree. (Sec. 4.1.5)

4.1.2 *Related Work*

The superset of the TCMC problem—TDMA-based convergecast—has been extensively surveyed in [IGK11], which classifies the scheduling algorithms/heuristics according to different assumptions, design objectives and design constraints.

TCMC scheduling has been studied in detail in a series of works by Zhang et al. [ZSJ09b, SZJ09, ZSJ09a, ZSJ13]. These provide the basis and starting point for my work. In particular, the optimal scheduling for the simple case of line and balanced binary tree is derived, and insights into the general case of arbitrarily structured trees are given. Important findings are as follows:

1. The *minimum length and channel scheduling* algorithms of line and balanced binary tree topologies are devised.

2. The minimum schedule length of the general tree topology is equal to $max\{2n_1 - 1, N\}$, where n_1 is the number of nodes in the largest subtree and N is the total number of nodes (excluding the GW).[1]

3. Given D (the tree depth) channels, a *minimum length scheduling* of general tree topology can be obtained with a polynomial time algorithm, even under the *single-packet-buffer assumption*, where each non-root node can store at most one packet.

4. The IP formulation of *minimum length scheduling* for general tree topology is proposed.

5. A scheduling heuristic is presented for general tree topology with channel count less than the tree depth D. The state-of-the-art heuristic produces near-optimal schedule length [2], but has a high complexity. Initially, it needs *packet priority assignment* as preparation; to schedule each slot, it performs 3 steps one by one: *schedule sink children*, *connectivity keeping* and *priority schedule*. I will show that my *busy-sender-first* heuristic is much simpler, requiring only a sort on the candidate transmissions for the scheduling of a slot, but performs better in terms of shorter scheduling length and less buffer consumption.

6. Various theoretical bounds on scheduling length and channels are given for general tree topology.

Optimal convergecast scheduling of a tree network with an arbitrary interference graph, by using one channel and spatial reuse is proved to be NP-hard as the *graph coloring* problem can reduce to it [EV10]. However, to my knowledge, whether the TCMC scheduling is NP-hard or not is still unknown. Therefore, I need to rely on heuristics for TCMC scheduling when the number of channels is between 2 and $D - 1$ (both ends inclusive).

1 N and n_1 are used with the same meanings throughout Sec. 4.1.

2 I reimplement the heuristic based on the current Matlab code provided by the authors, which is quite different from the description in [ZSJ09a].

Due to the NP-hardness of scheduling a tree topology network with arbitrary interference graph and one channel, [EV10] employs two coloring heuristics based on node-coloring and level-coloring. It is reported that the latter performs better when the network has higher density of packets at higher levels; otherwise the former performs better. I adapt their two coloring methods to my TCMC setting and propose two heuristics correspondingly — the *node-coloring* and the *level-coloring* heuristics. In my evaluation, the latter heuristic always outperforms the former.

4.1.3 *Optimal Scheduling with Integer Programming*

In this section, I recapitulate the optimal solution to *minimum length scheduling* based on IP [SZJ09, ZSJ09a, ZSJ13]. Then I propose another formulation of the same problem without using the auxiliary variables. Finally, I put forth my extension of the solutions to the *minimum length and buffer size scheduling* as well as the *minimum length and channel scheduling* problems.

4.1.3.1 *Common Constraints of TCMC Schedulings*

The following constraints and their corresponding expressions in inequalities are common to the TCMC scheduling with different objectives. $p_t(u)$ denotes the number of packets buffered at node u at the beginning of slot t. $s_t(u,v)$ denotes the scheduling of a link (u,v) in slot t; $s_t(u,v) = 1$ if the link is scheduled, otherwise $s_t(u,v) = 0$. To formulate the problem, I need an upper bound T for the schedule length, which can be obtained based on analysis or any scheduling heuristic (not necessary optimal). I assume that the GW has the node ID 0.

1. Begin and end conditions: S is the set of all source nodes. At the start, there is no packet at the GW and one packet at each $u \in S$. At the end, the GW has collected all the packets and all other nodes have empty buffer.

$$\begin{aligned} p_0(0) = 0, \quad p_0(u) = 1, \quad \forall u \in S \\ p_T(0) = |S|, \quad p_T(u) = 0, \quad \forall u \neq 0 \end{aligned} \tag{4.1}$$

2. Primary conflicts at non-leaf nodes: as the secondary conflicts, i.e., those caused by the links which share no common nodes but interfere with each other if scheduled simultaneously on the same channel, is eliminated because no spatial reuse is allowed (4.4), I only need to resolve the primary conflicts at any non-leaf node u. In (4.2), C_u denotes the set of the children of u. $f(u)$ is the parent of u.

$$0 \leqslant \sum_{v \in C_u} s_t(v,u) + s_t(u, f(u)) \leqslant 1, \quad \forall \text{ non-leaf node } u, \forall t \tag{4.2}$$

3. Change of buffer size after each timeslot: the change of the buffer size at a node u after timeslot t is subject to the *conservation of packets*.

$$p_t(u) = p_{t-1}(u) + \sum_{v \in C_u} s_t(v,u) - s_t(u, f(u)), \quad \forall u, \forall t \in \{1, ..., T\} \tag{4.3}$$

4. Constraints of orthogonal channels and buffer size: there are at most C transmissions in a timeslot, where C is the total number of channels. Also the maximum

number of packets that can be buffered at a node (except GW) is constrained by a value Q. Note that, in the case of single-packet-buffer ($Q = 1$), the IP degenerates into Binary Integer Programming (BIP) as all variables p_t, s_t only take value of 0 or 1. It is normally much more efficient than the corresponding IP problem, as efficient methods for solving BIP are available. However, general BIP problems are still NP-hard.

$$\begin{aligned} \sum_{(u,v)} s_t(u,v) &\leqslant C, \quad \forall \text{ links } (u,v), \forall t \in \{1,...,T\} \\ p_t(u) &\leqslant Q, \qquad\qquad \forall u \neq 0, \forall t \end{aligned} \tag{4.4}$$

4.1.3.2 *Scheduling Objectives Expressed as Optimization Goals*

For the *minimum length scheduling*, an IP solution is proposed in [SZJ09]. It introduces auxiliary variables $z_t \in \{0,1\}$, which are 1 or 0 depending on whether there are transmissions in slot t. The optimization is expressed in (4.5). Using the auxiliary variables z_t, the minimization ensures there are no *idle* slots.

$$\begin{aligned} &\text{minimize} \sum_{t=1}^{T} t z_t \\ &\text{s.t. constraints } 4.1, 4.2, 4.3, 4.4, s_t(u,v) \leqslant z_t, \forall \text{ link } (u,v). \end{aligned} \tag{4.5}$$

The same scheduling objective can also be expressed without z_t and the corresponding constraints of $s_t(u,v) \leqslant z_t$:

$$\text{minimize} \sum_{t=1}^{T} \sum_{u \in children(GW)} 2^t s_t(u,0) \tag{4.6}$$

Theorem 1 (The optimization of Eq. 4.6 gives a minimum length scheduling)**.** *If a scheduling satisfies the optimization goal (4.6), it is a* minimum length scheduling.

Proof. I prove by contradiction. Suppose there is another scheduling which uses T' slots where $T' < T$ and $\sum_{t=1}^{T'} \sum_u 2^t s'_t(u,0) \geqslant \sum_{t=1}^{T} \sum_u 2^t s_t(u,0)$, then

$$\sum_{t=1}^{T} \sum_u 2^t s_t(u,0) \leqslant \sum_{t=1}^{T'} \sum_u 2^t s'_t(u,0) = \sum_{t=1}^{T'} 2^t \sum_u s'_t(u,0) \leqslant \sum_{t=1}^{T'} 2^t = 2^{T'+1} - 2$$

In addition, any scheduling must end with transmitting a packet to the GW, otherwise, there are packets buffered in the network and the scheduling is incomplete. Therefore, $2^T \leqslant \sum_{t=1}^{T} \sum_u 2^t s_t(u,0) \leqslant 2^{T'+1} - 2$. This contradicts the supposition $T' < T$. □

Practically the coefficient 2^t in (4.6) gets large too quickly for an optimization software to perform IP correctly when it schedules a tree with many nodes.

I extend (4.5) to the IP solution of *minimum length and buffer size scheduling*. It serves as the optimal baseline for evaluating my heuristics. I introduce one more variable p denoting the maximum buffer size consumed. Obviously I have $1 \leqslant p \leqslant n_1$, n_1 being the number of nodes in the largest subtree, thus the optimization goal first prefers

a shorter schedule length and only under the same schedule length will it prefer a smaller maximum (worst-case) buffer size.

$$\begin{aligned} &\text{minimize} \sum_{t=1}^{T} n_1 \cdot t \cdot z_t + p \\ &\text{s.t. in addition } p_t(u) \leqslant p,\ \forall t \in \{0, ..., T-1\},\ \forall u \neq 0 \end{aligned} \tag{4.7}$$

The formulation for *minimum length and channel scheduling* is obtained by setting T to the minimum length $max\{2n_1 - 1, N\}$ (this is the minimum length when given enough channels [SZJ09, ZSJ09a, ZSJ13]) and making the number of channels C a variable. Then obviously the optimization goal is simply:

$$\text{minimize } C \tag{4.8}$$

4.1.4 *Suboptimal Scheduling with Heuristics*

Although the IP scheduling gives the desirable optimal solution, its computation time goes up exponentially, since IP is an NP-hard problem. Thus, it is only applicable when it is carried out infrequently and the network has a very small number of nodes. To schedule a relatively large network, or to stay adaptive to the quick change of routing topology, I have to rely on suboptimal heuristics, which run in polynomial time, but do not guarantee the solution optimality. I will present four heuristics — one originates from the famous *critical path scheduling* [KW59]; two are adapted from the coloring-based heuristics for single-channel convergecast scheduling in [EV10]; finally, the last one with the best performance comes from the intuition that the hotspot nodes with more transmissions left and more conflicts with other transmissions should be prioritized. Later, I extensively evaluate these heuristics together versus the state-of-the-art heuristic of [ZSJ09a, ZSJ13] and the optimal IP scheduling.

4.1.4.1 *The General Framework for Convergecast Scheduling*

The convergecast scheduling can be viewed as a decision problem: it has the begin and the end *states* described in (4.1). A *state* S is a vector containing the *buffer state* $p(n_i)$ of each node n_i. A node's *buffer state* is simply the number of packets buffered irrespective of the contents of these packets as they makes no difference to the scheduling algorithms. Specifically, $S = \left[p(n_1), p(n_2), ..., p(n_N)\right]$ where the tree has N nodes. That is to say that two *states* are equal if and only if every node has the same number of buffered packets. The scheduling in a timeslot transitions the system from one *state* to another. At any state S, there are a number of possible state transitions, determined solely by the *state* S (Markovian property). A scheduling can be viewed as a transitional sequence starting from the begin state S_b and finishing with the end state S_e (Fig. 4.1). Because each transition has the cost of one slot, a *minimum length scheduling* is a shortest path from S_b to S_e. However, as the number of states can be exponential, such a path is not trivial to find. Suppose the distances of nodes $n_1, n_2, ..., n_N$ to the GW are $h_1, h_2, ..., h_N$, if each node has one packet for delivery in each superframe, the number of all states is no more than $\prod_{i=1}^{N}(h_i + 1)$, the total number of states taking into account the position of each packet.

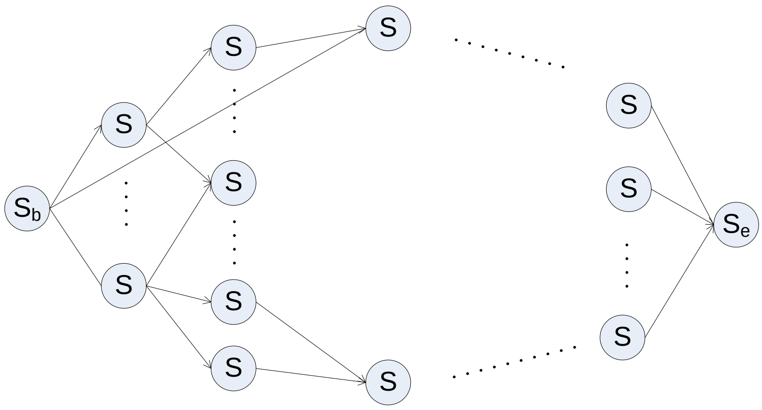


Figure 4.1: Convergecast scheduling as a decision problem. Each transition corresponds to the scheduling in a timeslot.

Viewing the convergecast as a decision problem leads me to the creation of a general framework for it (Alg. 1). Any two schedulings only differ in how they build the *candidate set* Q_t (nodes qualified for scheduling) and choose the *schedule set* S_t (nodes being scheduled, $S_t \subseteq Q_t$).

Algorithm 1: The convergecast scheduling framework.

Data: Convergecast scheduling problem
Result: A feasible schedule
fill the buffer of each node with the generated packets;
`// schedule timeslot t`
$t = 1$;
while *not all packets have arrived at the GW* **do**
    build *candidate set* Q_t from the nodes with at least one packet in the buffer and satisfying all additional constraints;
    `// select` $S_t \subseteq Q_t$ `subject to all scheduling constraints`
    schedule *schedule set* $S_t = \mathrm{select\_schedule\_set}(Q_t)$;
    update buffer size of the sender and receiver nodes;
    $t = t + 1$;

Wireless sensor nodes are normally memory-constrained, thus it makes sense to discuss about scheduling under buffer size constraint. I explore two variants: a) *single-packet-buffer*, and b) *multi-packet-buffer*. Nodes of the former variant can contain no more than one packet while nodes of the latter have unlimited buffer size. By properly building Q_t, a scheduling heuristic is applicable to any buffer constraint without further modification. In the single-packet-buffer case, Q_t consists of any node with non-empty buffer whose parent must have empty buffer. In the latter case, simply all nodes with non-empty buffer are included. In the evaluation, I find this significantly affects the runtime of heuristics. It runs faster under the *single-packet-buffer* setting, because the size of Q_t is normally much smaller.

4.1.4.2 *Heuristics for TCMC Scheduling*

In this section, I describe my four heuristics for TCMC scheduling in detail: *max-distance-first*, *node-coloring*, *level-coloring* and *busy-sender-first*. They are equally applicable if some sensor nodes are pure relays.

MAX-DISTANCE-FIRST HEURISTIC The idea is the same as the famous *critical path scheduling* [KW59], which always schedules the task that heads the longest processing chain. Thus, *max-distance-first* will transmit the packet farthest from the GW whenever possible, given a transmission chance (Alg. 2). Note $\mathtt{hops}(n_i)$ is the hop distance of node n_i to GW.

Algorithm 2: *max-distance-first*: $S_t = \mathtt{select\_schedule\_set}(Q_t)$.

1 sort Q_t in the decreasing order of $\mathtt{hops}(n_i)$;
2 select a set S_t of non-conflicting nodes in the order of the sorted Q_t until S_t is full or no more non-conflicting node is available;

Given N nodes, each with a packet to send, the worst-case total transmission times of convergecast among all possible spanning trees is $\frac{(N+1)N}{2}$, being $O(N^2)$, when the tree is a line. With a constant number of channels, the schedule length is also $O(N^2)$. Since Q_t has size $O(N)$, the sort takes time of $O(N \cdot \log N)$. In total, the *max-distance-first* has the time complexity $O(N^3 \cdot \log N)$.

NODE-COLORING HEURISTIC Inspired by the node coloring idea from [EV10], the heuristic first assigns different colors to nodes in primary conflict, then it schedules nodes with different colors in different timeslots circularly. In each timeslot, besides the nodes of a certain color, all other non-conflicting nodes are scheduled greedily as well. [EV10] suggested to color the nodes in the non-increasing order of node degree. I want to point out that assigning color to nodes to eliminate secondary conflicts is actually an edge-coloring problem. According to Vizing's theorem [Die05], the minimum number of colors to edge-color a simple graph G is either $\Delta(G)$ or $\Delta(G)+1$ where $\Delta(G)$ is the maximum node degree of G. The colors required for a tree topology is exactly $\Delta(G)$ because this is the case for any bipartite graph following König's theorem [Die05]. The coloring can be performed by doing a level-based traversal on the tree and coloring the incoming links of every node encountered. To my TCMC scheduling, this means, with enough channels to eliminate secondary conflicts, each node in a tree can be scheduled once every Δ timeslots, no matter how big the tree is.

Yet enough channels are not always available. The *node-coloring* heuristic needs to restrict the number of nodes of the same color by the total number of channels during the coloring process (Alg. 3). Considering nodes with more primary conflicts first complies with the common practice of considering high-degree nodes first in coloring. After each node is assigned a color, the network is ready to be scheduled with Alg. 4. The complexity of the first for loop in Alg. 4 is $O(N)$ as the loop checks no more than N nodes. Thus, the time complexity of the whole heuristic is $O(N^2 \cdot (N + N \cdot \log N)) = O(N^3 \cdot \log N)$. Assume M is the total number of colors, the upper bound of the scheduling length is $M \cdot N$ because a superslot contains at most M timeslots and in each superslot, at least one packet is delivered to the GW.

Algorithm 3: *node-coloring*: node coloring.

Data: C : total number of channels available, T: the routing tree
Result: Each node is assigned a color
`// tree pruning`
do post-order traversal on T, compute the *expected total transmission times* of each node, prune the node and the related link if the value is 0;
// $conflicts(n_i) = |\{children(n_i)\} \cup \{parent(n_i)\} \cup \{siblings(n_i)\} \setminus \{GW\}|$
sort all nodes in decreasing order of $conflicts(n_i)$ and increasing order of $hops(n_i)$ for each node n_i;
for *each node* n_i **do**
    $c = 1$;
    while *assigning color* c *to node* n_i *causes primary conflict or number of nodes with color* $c > C$ **do**
        $c = c + 1$
    assign c to node n_i;
$c = 0$;
M is the total number of colors used;

Algorithm 4: *node-coloring*: $S_t = select\_schedule\_set(Q_t)$.

for $i = 1$ **to** M **do**
    `// increase color c circularly, M is the total number of colors`
    $c = (c \mod M) + 1$;
    // $nodes_c$ `denotes the set of nodes in` Q_t `with color c`
    if $nodes_c$ *not empty* **then**
        add $nodes_c$ to S_t;
        break;
// $nodes_{oc}$ `denotes nodes in` Q_t `with a differnt color than c`
sort $nodes_{oc}$ in increasing order of $hops(n_i)$, decreasing order of $degree(n_i)$;
for *each node* $n_i \in nodes_{oc}$ *and scheduling it leads to no conflict and* $|S_t| \leqslant C$ **do**
    add n_i to S_t;

LEVEL-COLORING HEURISTIC Inspired by the level coloring idea from the same work [EV10], the heuristic first performs color assignment to levels. In the TCMC setting, if I eliminate the secondary conflicts, a tree level only conflicts with the two neighbouring levels. Therefore the minimum total number of colors M can be computed with Eq. 4.9. I can prove the minimum as no more than C levels of the same color is allowed and the neighbouring levels cannot be assigned the same color. Alg. 5 performs the coloring. Similarly, a superslot is composed of a sequence of timeslots of each possible color. In a timeslot of color c, I attempt to select a node from each level of color c. Additional nodes are added as long as the resulting set is non-conflicting and smaller than C (Alg. 6). Same as *node-coloring*, the time complexity of *level-coloring* heuristic is $O(N^3 \cdot \log N)$ and the upper bound of the scheduling length is $M \cdot N$.

$$M = \begin{cases} 1 & \text{if } D = 1 \\ 2 & \text{if } 2 \leqslant D \leqslant C \\ \lceil \frac{D}{C} \rceil & \text{if } D > C \end{cases} \tag{4.9}$$

Algorithm 5: *level-coloring*: level coloring.

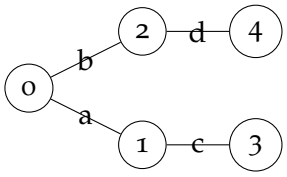
Data: C : total number of channels available, T: the routing tree
Result: Each level is assigned a color

```
prune tree T as in the node-based heuristic;
// D is the tree depth after pruning
// M is the total number of colors computed by Eq. 4.9
for d = 1 to D do
    assign color c = [(d − 1) mod M] + 1 to level d;
c = 0;
```

Algorithm 6: *level-coloring*: $S_t = \text{select\_schedule\_set}(Q_t)$.

```
for i = 1 to M do
    c = (c mod M) + 1;
    nodes_c = select one node from each level of color c and in Q_t. In the same
    level, a node with a higher node degree is preferred;
    if nodes_c not empty then
        add nodes_c to S_t;
        break;
sort nodes_o (other nodes in Q_t) in increasing order of hops(n_i) and decreasing
order of degree(n_i);
for each node n_i ∈ nodes_o and scheduling it leads to no conflict and |S_t| ⩽ C do
    add n_i to S_t;
```

BUSY-SENDER-FIRST HEURISTIC Though *max-distance-first* heuristic has the clear intuition of prioritizing packets farther away from the GW, such "fairness" may not give the *minimum schedule length*. It is more important to evacuate the data as fast



max-distance-first					*busy-sender-first*			
c	a	a	b	b	a	b	a	b
d					d	c		

Figure 4.2: A sample tree topology, for which *max-distance-first* performs sub-optimally.

as possible by parallelizing as many transmissions as possible. For this purpose, I find it is very effective to prioritize the *busy sender*, the node in Q_t with the largest number of remaining transmissions. Such scheduling has the intuition to relieve any hot-spots, which may become the bottleneck and harm the parallelism as they normally cause more conflicts with transmissions of other nodes. Fig. 4.2 gives an example of a simple topology, with 2 channels, 5 timeslots are required by *max-distance-first*, however, if I prioritize nodes with more *unscheduled transmissions* by applying *busy-sender-first*, 4 timeslots are enough. Different from the former heuristics, *busy-sender-first* uses *dynamic* information of a node: 1) the unscheduled transmissions of a node, 2) the sum of the unscheduled transmissions of all conflicting nodes. Besides, priority is given to high level nodes in order to prevent a *gap*, i.e., the lower levels having empty buffers while high levels having non-empty buffers. The scheduling heuristic is listed in Alg. 7. In the listing, $\mathtt{tx\_unscheduled}(n_i)$ denotes node n_i's unscheduled transmissions; $\mathtt{conflict\_tx\_unscheduled}(n_i)$ denotes the sum of the unscheduled transmissions of all conflicting nodes ($\mathtt{conflict\_nodes}(n_i) = \mathtt{children}(n_i) \cup \mathtt{parent}(n_i) \cup \mathtt{siblings}(n_i) \setminus \{GW\}$).

$$\mathtt{conflict\_tx\_unscheduled}(n_i) = \sum_{n \in \mathtt{conflict\_nodes}(n_i)} \mathtt{tx\_unscheduled}(n)$$

Same as *max-distance-first*, this heuristic has the time complexity $O(N^3 \cdot \log N)$. The heuristic is also directly applicable to any mesh topology without further modification.

Algorithm 7: *busy-sender-first*: $S_t = \mathtt{select\_schedule\_set}(Q_t)$.

1 sort Q_t in the decreasing order of $\mathtt{tx\_unscheduled}(n_i)$, decreasing order of $\mathtt{conflict\_tx\_unscheduled}(n_i)$ and decreasing order of $\mathtt{hops}(n_i)$;
2 select a set S_t of non-conflicting nodes in the order of the sorted Q_t until S_t is full or no more non-conflicting node is available;

4.1.5 *Evaluation*

In this section, I evaluate the four heuristics proposed and the state-of-the-art *time- and channel-optimal convergecast* heuristic from [ZSJ09a, ZSJ13] (henceforth, I call it *convergecast* heuristic for short), as well as the IP-based optimal scheduling. The evaluation focuses on two metrics: *scheduling length* and *maximum buffer size*. Furthermore, I also explore the tradeoff between number of channels and schedule length, as well as the impact of topology on schedule length. All heuristics and the IP scheduling are implemented in Java. In addition, I use the MOSEK Optimization Library [MOS] for solving IP problems.

4.1.5.1 *Evaluation of Heuristics and IP Scheduling*

I carried out two experiments: *experiment A* evaluates the heuristics in comparison with the optimal IP solution for small trees ($\leqslant$ 30 nodes) and *experiment B* evaluates the heuristics with small to large trees (2 to 1024 nodes).

The optimal solution is obtained by solving the IP-based *minimum length and buffer size scheduling*. As IP is generally NP-hard, I only evaluate random trees of limited size in *experiment A*, i.e., with 5 to 30 nodes. For each size, I generate 10 random trees, on which I run the 5 heuristics and the IP scheduling. To better evaluate the heuristic with increasing number of nodes, I carry out more extensive evaluation in *experiment B*, with trees of $2^n, n \in \{1, 2, ..., 10\}$ nodes. For each tree size, I randomly generate 50 trees, on which I run the 5 heuristics, but not the IP scheduling, since it would be computationally too expensive. In both experiments, for each generated tree, the heuristics and the IP scheduling run under the setting of different number of channels (2 to D, the tree depth) and two buffer constraints (single-packet-buffer and multi-packet-buffer).

SCHEDULING LENGTH In *experiment A*, the IP scheduling provides the *minimum schedule length*, to which the performance of the various heuristics is normalized. In *experiment B*, as the computation time of IP scheduling is too long for large trees, instead of normalizing the schedule length of a heuristic to the *minimum schedule length*, I normalize it to the *schedule length lower bound*. Given channel count and single-/multi-packet-buffer setting, the tight *schedule length lower bound* can be obtained with the Corollary 4 in [ZSJ09a]. Consequently, the normalized schedule length may be overestimated and the actual performance of heuristics can only be better than is shown in the plots.

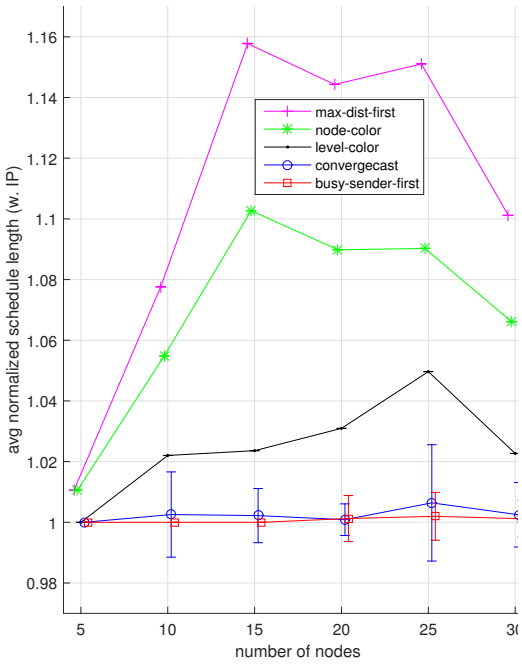
From Fig. 4.3 and the experimental results in detail I can observe the following points.

1. Generally, in both experiments, each heuristic performs consistently: *max-distance-first*, *node-coloring*, *level-coloring*, *convergecast* and *busy-sender-first*, in the order of increasing performance. The only exception is in *experiment B*, where the *level-coloring* is a bit better than *convergecast* when the trees are over 512 nodes, which suggests giving equal scheduling chance to nodes at different levels may be beneficial for large trees.

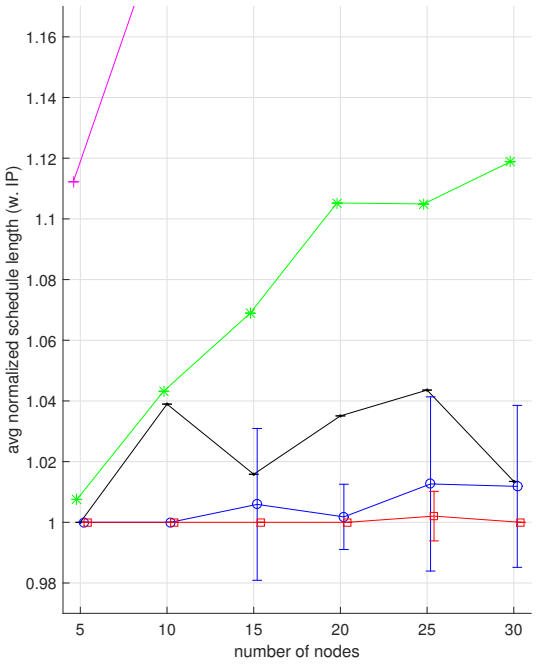
2. The average schedule lengths of the last three heuristics always satisfactorily stay within 5% of the *minimum length* or the *lower bound*, qualifying them to be used in real-world scheduling; *max-distance-first* has the worst average schedule length performance, but it is still within 45% of the optimum (part of the plots are not shown due to space limit).

3. The relative difference in performance is bigger under the multi-packet-buffer setting. This is because the single-packet-buffer setting is more restrictive and gives less freedom in selecting nodes to be scheduled.

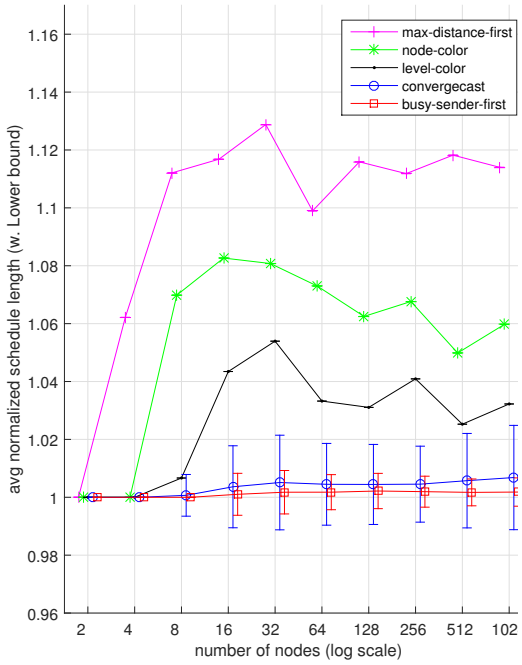
4. My unique *busy-sender-first* heuristic performs significantly better than all the others, including the state-of-the-art *convergecast*, with the smallest average result, smallest worst-case result and smallest standard deviation in all cases. Its



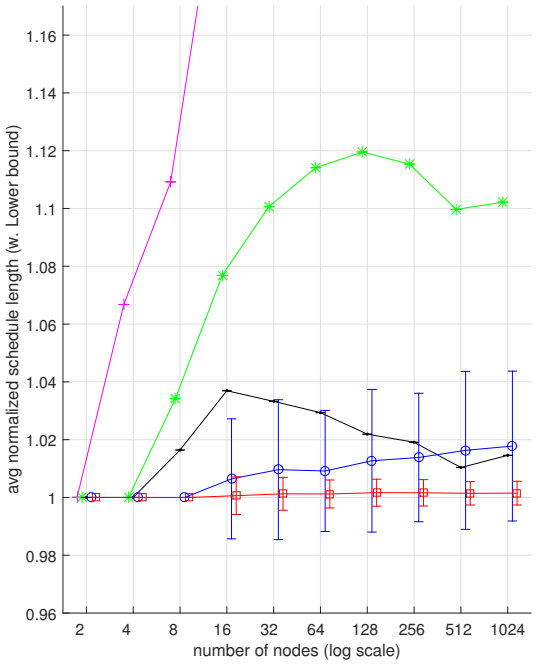
(a) Experiment A: single-packet-buffer



(b) Experiment A: multi-packet-buffer



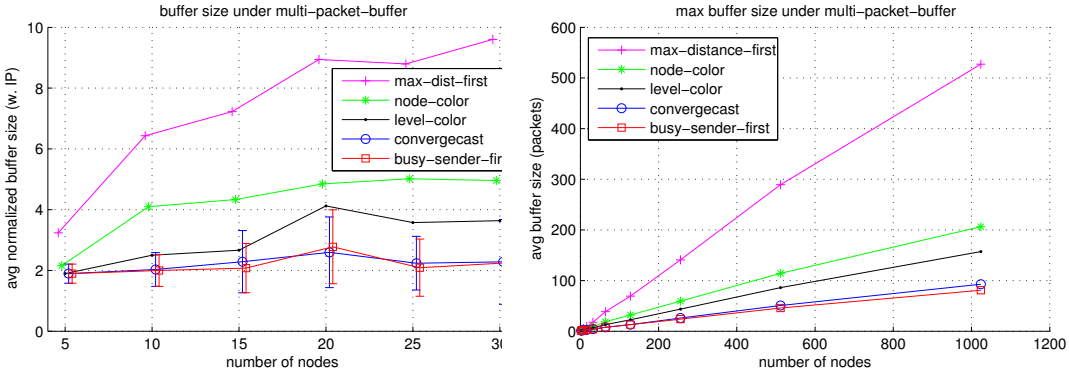
(c) Experiment B: single-packet-buffer



(d) Experiment B: multi-packet-buffer

Figure 4.3: Schedule length of *experiment A* and *B*. In (a) and (b), the results are normalized to those of the optimal IP scheduling; in (c) and (d), they are normalized to the *schedule length lower bounds*. The standard deviations are shown for the two best heuristics for better comparison, but are omitted for the other three heuristics for clearer plotting. Curves are moved a bit to make the errorbars discernible.

average performance is always within 1.22% of the *minimum length* or the *lower bound* while the performance of *convergecast* stays only within 1.78%. And the performance advantage of the *busy-sender-first* in comparison to the second-best *convergecast* is increasing with number of nodes (Fig 4.3(c),(d)). Taking a closer look into the schedules of *experiment A*, I find that *busy-sender-first* attains the minimum length for 98% of all schedules, and for the other 2% cases, it uses only one more timeslot. Also the worst-case performance of *busy-sender-first* is far better than others: at worst *busy-sender-first* uses 4.6% and 8.3% more timeslots than the references in *experiment A* and *B*, while the second-best *convergecast* uses 13% and 21.8% more timeslots. These results unambiguously displays *busy-sender-first*'s excellent performance in schedule length as a heuristic.



(a) Experiment A: multi-packet-buffer

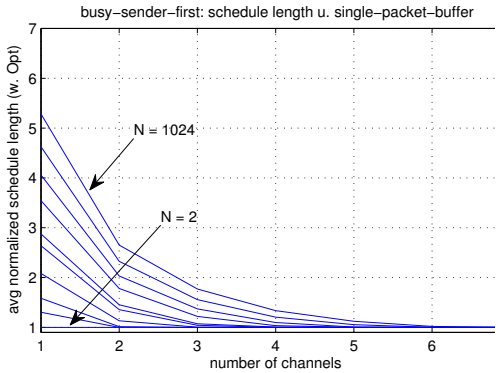
(b) Experiment B: multi-packet-buffer

Figure 4.4: Maximum buffer size of *experiment A* and *B*. In (a), the results are normalized to those of the optimal IP scheduling; in (b), they are in the unit of packets. The curves in (a) are moved a bit to make the errorbars discernible.

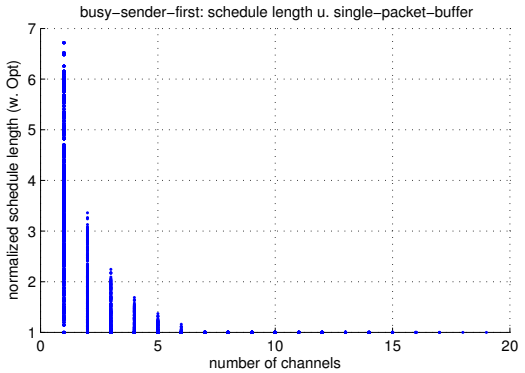
MAXIMUM BUFFER SIZE Only the results under the multi-packet-buffer setting are presented, as under the single-packet-buffer setting, the *maximum buffer size* is always 1. In *experiment A*, the IP scheduling provides the minimum reference, to which the performance of all heuristics are normalized; in *experiment B*, as it is impossible to get the optimum *maximum buffer size* in reasonable time and there is no derivable lower bound for it, I display the results in raw values of packets. I can observe from Fig. 4.4 the following points.

1. The performance of memory consumption generally keeps the same order as the scheduling length, i.e., *max-distance-first*, *node-coloring*, *level-coloring*, *convergecast* and *busy-sender-first* in increasing order. But in *experiment A* (Fig. 4.4(a)), the average memory consumption and standard deviation show that the performance of *busy-sender-first* and *convergecast* are similar.

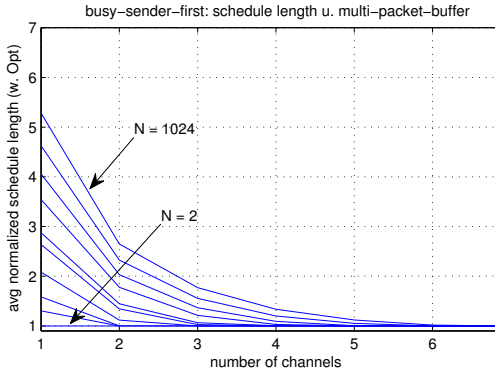
2. As confirmed by both experiments, the memory consumption of *max-distance-first* goes up fastest with the number of nodes. The reason is that generally packets in high levels are prioritized, leading to the fact that memory consumption at a GW child is statistically proportional to the number of nodes in the corresponding subtree (the curve of *max-distance-first* in Fig. 4.4(b) is almost linear).



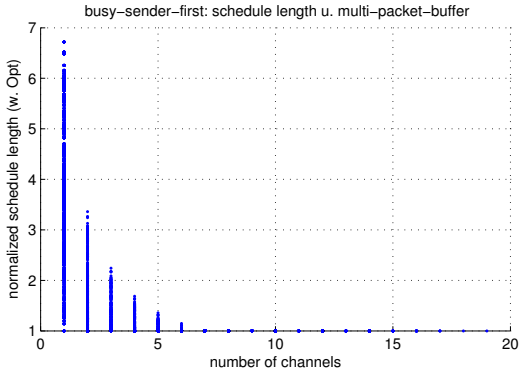
(a) average performance, single-packet-buffer



(b) scatterplot of all samples, single-packet-buffer



(c) average performance, multi-packet-buffer



(d) scatterplot of all samples, multi-packet-buffer

Figure 4.5: The tradeoff between channels and schedule length under single-packet buffer and multi-packet-buffer setting. Schedule length is normalized to the optimum length $\max\{2n_1 - 1, N\}$.

3. The curve of *busy-sender-first* in Fig. 4.4(b) is the flattest among all, which means that with the increase of nodes the *maximum buffer size* goes up more slowly than the others. For example, at 2 nodes, *busy-sender-first* needs on average half the buffer size of *max-distance-first* and the same amount as *convergecast*, while at 1024 nodes, it only needs 15% of the former and 87% of the latter.

4. Although *busy-sender-first* performs the best, it still uses more than twice of the optimal buffer size in *experiment A*. In contrast to *schedule length*, there seems to be bigger room for improvement regarding memory consumption.

On average, the *busy-sender-first* heuristic needs a buffer size of about 46 packets for a network with 512 nodes — a sufficient upper bound of the scale of a network for the purpose of industrial automation. Since a packet is no more than 127 bytes, this takes less than 6K bytes which can be fit into the 10K bytes Random Access Memory (RAM) of the most widely used TelosB mote [Moto4].

4.1.5.2 *Tradeoff between Number of Channels and Schedule Length*

The available wireless channels are very often a more stringent and precious resource than buffer space. With the experiment results of the *busy-sender-first* in *experiment B*, I evaluate the tradeoff between number of channels and schedule length. I normalize

the schedule length to the optimum value $max\{2n_1 - 1, N\}$, the shortest schedule length possible given enough channels. In Fig. 4.5(a) and (c), I show the tradeoff between average schedule length and number of channels of each tree size under the single-packet-buffer and the multi-packet-buffer setting respectively (the 10 lines from top to down correspond to 1024 to 2 nodes respectively). In Fig. 4.5(b) and (d), I show scatterplots that capture the dataset of all (number of channels, schedule length) pairs of all tree sizes. The results for two buffer settings are almost the same. As expected, the marginal improvement of schedule length decreases with the number of channels. On average, starting with 1 channel, 2 channels almost halves the schedule length while 3 channels further decreases the schedule length to 1/3. On average, even with a big network of 1024 nodes, 3 channels already decrease the schedule length from 5.28 times to 1.77 times of the optimum while the worst-case decreases from 6.72 times to 2.24 times. With 7 channels, for all evaluated trees with up to 1024 nodes, I always obtain the optimum schedule length, which shows that the 16 orthogonal channels specified by the IEEE 802.15.4 standard is sufficient for the scale of a control network. In general, for a network with no more than 1024 nodes, given 3 to 4 channels will push the schedule length below twice the optimum. More channels will further reduce the length, but the marginal improvement decreases.

4.1.5.3 *The Impact of Tree Topology on Scheduling Length*

The configuration of tree topology has a great impact on scheduling length. For instance, given a network of N nodes and D channels, if it is balanced, the optimal schedule length is exactly N; if the GW has only one subtree, then the optimal schedule length is $2N - 1$. The difference is almost a factor of two. If less than D channels are available, according to the Corollary 4 of [ZSJ09a], the schedule length lower bound is $L \geqslant \lceil \frac{\overline{D} \cdot N}{C} + \frac{C}{2} - \frac{1}{2} \rceil$, where C is the number of channels and $\overline{D}$ is the average depth of all nodes, assuming multi-packet-buffer. $\overline{D}_{max} = \frac{N+1}{2}$ if the network is a line and $\overline{D}_{min} = 1$ if all nodes are one hop away from the GW. Thus the difference in schedule length can be as big as $\frac{N+1}{2}$ times.

Table 4.1: Quality of various hypotheses about topology choice. The left values are the performance on the learning dataset of 16-node trees. The right values are the performance on the test dataset of 15-node trees. "positive", "negative" and "neutral" means a scheduling confirms, rejects or neither confirms nor rejects a hypothesis.

Hypothesis (tree a and tree b)	positive %	negative %	neutral %
1: $\beta_a < \beta_b \Rightarrow L_a < L_b$	63.9 | 64.6	12.6 | 11.4	23.5 | 24.0
2: $\overline{D}_a < \overline{D}_b \Rightarrow L_a < L_b$	93.1 | 91.9	1.6 | 2.1	5.3 | 6.0
3: $D_a < D_b \Rightarrow L_a < L_b$	71.5 | 70.4	8.4 | 8.5	20.1 | 21.1
4: $\overline{D}_a < \overline{D}_b \Rightarrow L_a < L_b$ else $D_a < D_b \Rightarrow L_a < L_b$ else $\beta_a < \beta_b \Rightarrow L_a < L_b$	94.6 | 93.6	1.7 | 2.1	3.7 | 4.3

To learn the impact of the configuration of the topology on the schedule length, I enumerate all rooted trees of 16 nodes with the Beyer-Hedetmieni algorithm [BH80]. These 634,847 trees encompass all possible tree topologies. I further restrict the trees

to those whose maximum degree is no more than 4 because real-world WSN nodes can only have a limited number of links to other nodes (530,657 trees are qualified). Then I use *busy-sender-first* to schedule the trees under the condition of 2 channels and multi-packet-buffer. Three important topology metrics are investigated:

1. Balance Factor β: the percentage of nodes in the biggest subtree. $\beta = \frac{n_1}{N}$. $\beta \in [\frac{1}{N}, 1]$, where $\frac{1}{N}$ denotes a highly balanced tree — every node is directly connected with the GW; 1 denotes the unbalanced situation of the only subtree.

2. Average Depth $\overline{D}$: average depth of all nodes to the GW. It appears in the schedule length lower bound.

3. Tree Depth D: the tree depth.

Intuitively, smaller β, $\overline{D}$ and D would result in short schedule length. But what is the best combination of metrics? With the learning dataset of all qualified 16-node trees, I evaluate all combinations of the 3 metrics and find that my *Hypothesis 4* works best (Table 4.1 contain the hypotheses based on single metric and the best combination of the 3 metrics). The hypothesis is equivalent to the guideline in topology choice: given a number of nodes, I first choose a spanning tree with small $\overline{D}$; if $\overline{D}$ is the same, I prefer a tree with small D; if D is still the same, I choose a tree with small β, which is confirmed 94.6% and is rejected 1.7% of the times by the learning dataset. Moreover, the test dataset of all 15-node trees (197,306 trees) with the same aforementioned properties also confirms the superiority of this hypothesis (values on the right side in Table 4.1).

4.1.6 *Conclusion*

Applying WSN in industrial automation settings requires fast and reliable convergecast. I focus on the scheduling of tree-based multi-channel convergecast without spatial reuse (TCMC), a key component of WSNs, which is directly applicable to standards such as WirelessHART and ISA100.11a. I have extended the optimal IP-based solution to the *minimum length and buffer size scheduling* and the *minimum length and channel scheduling*. Because the optimal TCMC scheduling algorithm is not yet found, given a limited number of channels, scheduling has to rely on efficient heuristics. I have presented a unique perspective of viewing the TCMC scheduling as a decision problem and created a general and flexible framework for scheduling heuristics. Furthermore, I propose 4 heuristics, among which *busy-sender-first* attains a schedule length that is within 1.22% of the minimum length, significantly better than the state-of-the-art heuristic in [ZSJ09a, ZSJ13]. It also incurs slightly less memory consumption. Another big advantage of the *busy-sender-first* heuristic is its conceptual simplicity and the resulting simplicity in implementation, in comparison with the state-of-art heuristic. Besides, I evaluate the tradeoff between number of channels and scheduling length and give guidelines on the choice of number of channels. Last, I propose an effective method for choosing the configuration of the tree topology based on evaluation on all rooted trees of same number of nodes, which leads to short schedule length.

4.2 TDMA SCHEDULING FOR PERIODIC CONTROL SYSTEMS OF ARBITRARY TOPOLOGY

4.2.1 *Introduction*

Control systems are ubiquitous in industry and everyday life. They can be as simple as a thermostat that automatically adapts the room temperature or as complex as the European power network that coordinates the production and consumption of electricity continuously, as well as maintaining the safe operation of the large distributed system [ÅM08].

In this work, I focus on the scheduling of TDMA-based wireless sensor networks for industrial automation featuring *periodic control*. Example of periodic control systems are assembly lines for car manufacturing and the condition control (e.g., pressure, temperature) of a chemical reaction in order to obtain quality end-products efficiently. This type of system is composed of one or more *control loops*, each of which performs the *control* periodically on one *output variable* (e.g., temperature). In each period, a *control loop* performs *sensing*, *computation* and *actuation* with the goal of making the *real output* follow the *desired output* (Fig. 4.6).

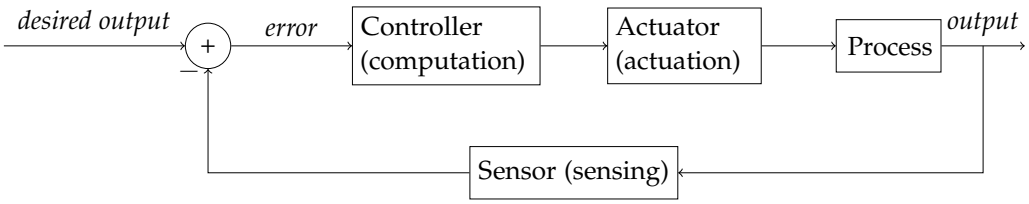


Figure 4.6: A control loop periodically performs sensing, computation and actuation.

Because periodic control systems are normally *hard real-time systems*, i.e., each activation of a sensing-computation-actuation loop must finish before a hard deadline, the network design must correspondingly guarantee hard end-to-end transmission latencies. Since the wireless communication has lower reliability, lower throughput and larger latency, compared to the wired communication, how to schedule the networks so that the hard deadlines and reliability of retransmissions are guaranteed at the same time is very challenging.

My work is based on a very general communication model of multi-rate periodic control systems. It allows arbitrary topology, multiple gateways, multiple flows with arbitrary period and deadline requirements where each flow can have multiple routing paths. Under such a model, I apply the *Single Controller Activation (SA) scheduling*, i.e., scheduling the controller-to-actuator transmissions after the sensor-to-controller transmissions, as I argue that it is more desirable than the *Multiple Controller Activation (MA) scheduling*, i.e., scheduling each sensor-to-actuator path independently. The main contributions of this work are as follows.

1. I propose a general framework for scheduling algorithms. Based on that, I implement a number of fixed and dynamic priority scheduling algorithms borrowed from multi-processor real-time scheduling and existing works on TDMA scheduling, adapted to my system model (Sec. 4.2.4).

2. Through extensive evaluation, I identify the best algorithm Least Laxity First (LLF) in terms of high schedulability rate, low execution time and low memory overhead for network operation (Sec. 4.2.5.2).

3. I design a simple *opportunistic aggregation* scheme that works seamlessly with any scheduling algorithm and I demonstrate that it significantly increases schedulability (Sec. 4.2.4.4 and Sec. 4.2.5.3).

4. I design a unique scheme called *repetitive scheduling* that works with any scheduling algorithm under the condition that the periods are harmonic. It is very effective for the case of implicit deadline (meaning deadline is equal to period) which incurs only minimal penalty on schedulability, but has the ideal property of highly scalable execution time and schedule table size (Sec. 4.2.4.5 and Sec. 4.2.5.4).

4.2.2 *Related Work*

The research that deals with a problem most similar to ours is a series of works by Saifullah et al [SXLC10, SXLC11a, SXLC11b]. They discuss the TDMA scheduling problem of WirelessHART networks with multiple channels and no spatial reuse. It assumes a network of arbitrary topology and one gateway. A number of flows (control loops) are to be scheduled where each flow corresponds to the transmissions from a node (sensor) to the gateway (controller) and then to another node (actuator). Each flow is activated periodically and has a hard deadline. [SXLC10] shows that a control loop can also have multiple paths by mapping each path to an independent flow, which needs multiple activation of the controller (MA scheduling). A model of multi-path control loop that is more realistic and offers higher transmission reliability is to schedule the controller-to-actuator transmissions after all sensor-to-controller transmissions, which only needs one activation of the controller (SA scheduling). I use this model and additionally, I allow more than one gateways for better performance.

The WirelessHART scheduling problem is proved to be NP-hard by [SXLC10]. It gives a branch-and-bound optimal scheduling algorithm but it is only feasible for networks of very small size. Then, [SXLC10] introduces the Conflict-aware Least Laxity First (CLLF) heuristic which has performance dominating the other algorithms. I have adapted CLLF to my problem setting and have evaluated it together with a set of other algorithms. I find that CLLF has incurred much higher computational overhead than the best algorithm LLF, despite delivering worse performance. I have performed an extensive evaluation on each scheduling algorithm by using both ideal and realistic link models, with varying number of flows, total utilization of the network and number of channels.

An end-to-end latency analysis of the fixed priority scheduling algorithms for the WirelessHART scheduling problem is given in [SXLC11a], which incorporates the most up-to-date advance in response time analysis. [SXLC11b] discusses the priority assignment for real-time flows. It proposes an optimal algorithm based on local search for any given worst case latency analysis and then provides an efficient heuristic.

4.2.3 *System Model*

This section describes the TDMA-based wireless sensor network model for multi-rate periodic control systems. Specifically, I extensively discuss the topology, the link quality, the communication, and the routing models associated with the system model.

4.2.3.1 *Topology Model*

The topology of a wireless network for a periodic control system is modeled as an *undirected simple graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of undirected links [3]. Furthermore, $\mathcal{V} = \mathcal{N} \cup \mathcal{M}$, where $\mathcal{N}$ is set of motes (a mote is a sensor, an actuator or a pure relay, all with the ability to relay packets), and $\mathcal{M}$ is the set of gateways. An element $e \in \mathcal{E}$ has a corresponding link quality (packet reception rate) between 0 and 1. I assume any two gateways are connected to each other with perfect link quality of 1 and zero latency. This is a simplification of the real-world scenario in which gateways are access points to the wired network infrastructure.

4.2.3.2 *Link Quality Model*

Simulating link quality in practice requires setting up a realistic radio propagation model. The so-called log-normal model is the preferred one due to its accuracy in the large-scale fading, which is typical in industrial factory environments [RM89, ZK04, TJV+08]. For each pair of nodes (at least one is non-gateway), I compute link quality in this way, and I keep a link only when the link quality is above 50%.

The *path loss* of log-normal model can be computed as follows:

$$PL(d) = PL(d_0) + 10\eta \cdot \log_{10}\left(\frac{d}{d_0}\right) + X_\sigma \tag{4.10}$$

where $PL(d)$ is the path loss of signal strength at distance d, $PL(d_0)$ is the path loss at the reference distance d_0, η is the path loss exponent and X_σ is a zero-mean Gaussian random variable with standard deviation σ. The term X_σ is due to the shadowing effect [Rap01]. I set reference distance $d_0 = 15\,m$, path loss at reference distance $PL(15) = 71.84\,dBm$, path loss exponent $\eta = 2.16$ and $\sigma = 8.13$, all obtained by the extensive real-world measurement in [TJV+08]. These are the average parameters of 2.4 GHz frequency in normal in-door factory environment for both cases of Line-of-sight (LOS) propagation and Non-line-of-sight (NLOS) propagation.

Thus the reception power $P(d)$ at distance d is

$$P(d) = P_t - PL(d) \tag{4.11}$$

where P_t is the transmission power. The typical IEEE 802.15.4 radio chip CC2420 [CC213] has P_t in the range of $-25\,dBm$ to $0\,dBm$. I choose $0\,dBm$ in my evaluation since it provides the highest packet reliability when spatial reuse is disabled. Under a clear environment, the *noise floor* of the CC2420 radio is about $-98\,dBm$. Hence, I can compute the Signal-to-noise Ratio (SNR) as:

$$\gamma = P(d) - P_n = P(d) + 98 \tag{4.12}$$

3 For simplicity, I use the undirected link model, but my work can be easily extended to the more realistic directed link model.

and Symbol Error Rate (SER) as:

$$SER = \frac{1}{2} erfc\left(\frac{\beta_1(\gamma - \beta_2)}{\sqrt{2}}\right) \tag{4.13}$$

by means of the empirically determined accurate TOSSIM model [LLWC03] where $\beta_1 = 0.9794$ and $\beta_2 = 2.3851$. Finally, I get the Packet Reception Rate (PRR)

$$PRR = (1 - SER)^{2l} \tag{4.14}$$

where l is the packet length measured in bytes. An IEEE 802.15.4 frame contains a preamble of 4 bytes, 1 byte Start Frame Delimiter (SFD), 1 byte of frame length and PHY Service Data Unit (PSDU) of variable length up to 127 bytes. I set l to 133 bytes in order to take into consideration the worst-case scenario.

4.2.3.3 *Communication Model*

I have opted for the TDMA protocol in my communication model since it provides deterministic latency. In order to minimize the internal interference due to concurrently scheduled links, I disable the spatial reuse in the same fashion as WirelessHART does. Assume that there are C orthogonal channels in total (e.g., for IEEE 802.15.4, $C = 16$), which means the number of concurrently scheduled links should not be greater than C. Although links between gateways are assumed to have zero latency and 100% link quality, the link between a gateway and a mote, or between two motes, is generally imperfect and it takes one slot to send a packet through such a link. Moreover, each gateway or mote is equipped with one half-duplex radio which cannot send and receive simultaneously.

I discuss the most common and simplest case in which a feedback loop is activated periodically and has Single Input and Single Output (SISO), i.e., one sensor and one actuator. [4] I call the communication of a loop a *flow*.

Because the TDMA protocol is applied, it is natural to model the communication as a discrete-time system. A unit time is set to be the duration of a slot. Denote the period of flow f as p, where p is a positive integer. The sensor samples at times $k \cdot p$, where $k = 0, 1, 2, ...$ Time $k \cdot p$ is the instant at which the flow f is activated for the kth time. As pointed out in [ZBP01], which analyzes a constant network-induced latency model, if the control laws are time-invariant, there is no need to differentiate the sensor-to-controller latency d^{sc} and the controller-to-actuator latency d^{ca}. Hence, I use the sum *flow latency* $d = d^{sc} + d^{ca}$ for analyzing communication schedulability. I abstract the latency requirement of flow f to be $d \leqslant d_{max}$, where d_{max} is the maximum allowable latency. [ZBP01] illustrates with an example that the system can tolerate a flow latency d larger than p while being still stable when the period p is small enough. However, when p gets large, the d_{max} gets smaller than p. It is determined by the analysis on the stability and performance of the feedback loop, and can be lower, equal or larger than the period p.

Additionally, *flow reliability* also needs to be guaranteed. To obtain stability and satisfactory performance of a feedback loop, the flow reliability r should be above

4 My scheduling algorithms can be easily extended to the Multiple Inputs and Multiple Outputs (MIMO) setting.

a lower bound r_{min}, i.e., $r \geqslant r_{min}$. For control applications, normally the reliability lower bound r_{min} does not need to be so high as in monitoring applications [ZBP01, MPFJ10]. Routing can influence both flow latency and flow reliability; however, tackling this issue is beyond the scope of this thesis.

4.2.3.4 *Routing Model*

I consider a multi-path scheme for modeling the routing of a flow to ensure flow reliability. That is, when the lower bound requirement of a flow, $r \geqslant r_{min}$, cannot be satisfied by a single sensor-to-actuator path (*sa-path*), then multi-path routing might sort this problem out. A flow f is split into two parts: one comprising those paths starting at the sensor and ending at an arbitrary gateway, and another one comprising those paths starting at an arbitrary gateway and ending at an actuator. Specifically, the flow f is divided into a *sensor-to-controller flow* f^{sc} (*sc-flow*) and the *controller-to-actuator flow* f^{ca} (*ca-flow*). The paths in f^{sc} are called *sc-paths*, and denoted by π_i^{sc}, where $i = 1, 2, ...,$; while the paths in f^{ca} are called *ca-paths*, and denoted by π_j^{ca}, where $j = 1, 2, ...,$. Note that a flow may have different number of sc-paths and ca-paths. When the sensor data arrives at a gateway through an sc-path, the control algorithm is ready to run. Once the control algorithm has finished execution, the output is sent to the actuator through a ca-path. The proposed routing model is general enough, and works in a variety of concrete situations.

My next step is to decide on a scheduling strategy based on the following aspects: flow reliability, control algorithm executions, and flow latency. Suppose flow f have n sa-paths $\pi_0, \pi_1, ..., \pi_{n-1}$. [5] Each sa-path π_i is composed of an sc-path π_i^{sc} and a ca-path π_i^{ca}. Note that the gateway end-points of the two paths may be different, since I assume full connection between gateways.

Now I consider two scheduling models: SA scheduling and MA scheduling. SA specifies that the control algorithm runs only once for each activation of the flow, after the transmissions on all sc-paths. Then the controller output is sent on all ca-paths. MA schedules independently different sa-paths, and the same control algorithm runs n times, as many times as there are paths. For each sa-path, the algorithm runs once the corresponding sc-path finishes transmissions.

I analyze the SA and MA models in terms of the three aforementioned issues to decide which of them is better.

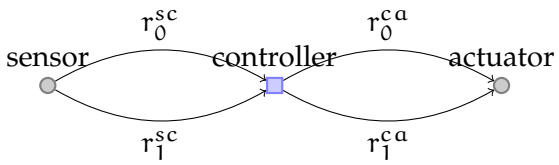


Figure 4.7: An end-to-end flow with two sa-paths π_0 and π_1, where r_i^{sc} and r_i^{ca} are the reliabilities of the sc-path and the ca-path of the sa-path π_i, respectively.

1. Flow reliability. Given a two-path routing such as the one in Fig. 4.7, the flow reliabilities of the models SA and MA are:

5 The concept of an sa-path is not really needed in the SA model and makes no sense when the number of sc-paths and ca-paths are different. I introduce it only for the sake of comparing the SA and MA models.

$$r^{SA} = [1-(1-r_0^{sc})(1-r_1^{sc})][1-(1-r_0^{ca})(1-r_1^{ca})]$$
$$r^{MA} = 1-(1-r_0^{sc}r_0^{ca})(1-r_1^{sc}r_1^{ca}) \tag{4.15}$$

The difference between the two reliabilities is:

$$r^{SA} - r^{MA} = r_0^{ca}r_1^{sc}(1-r_0^{sc})(1-r_1^{ca}) + r_0^{sc}r_1^{ca}(1-r_0^{ca})(1-r_1^{sc}) \geqslant 0 \tag{4.16}$$

This shows that the SA model provides equal or higher reliability in the case of two paths. The result can be extended to n paths as follows.

Theorem 2 (SA model has reliability greater than or equal to MA model)**.** *For a flow with $n \geqslant 1$ routing paths, the end-to-end reliability of the* SA *model is greater than or equal to that of the* MA *model, i.e., $r^{SA} \geqslant r^{MA}$.*

Proof. In the SA model, the probability that a sensor packet arrives at the controller is $1-\prod_{i=0}^{n-1}(1-r_i^{sc})$. This is also the probability of an actuator packet to be transmitted on each ca-path. Under the MA model, the probability of an actuator packet to be transmitted on the ith ca-path is r_i^{sc}. Since $r \in [0,1]$, $1-\prod_{i=0}^{n-1}(1-r_i^{sc}) \geqslant 1-(1-r_i^{sc}) = r_i^{sc}$. Therefore, $r^{SA} \geqslant r^{MA}$. □

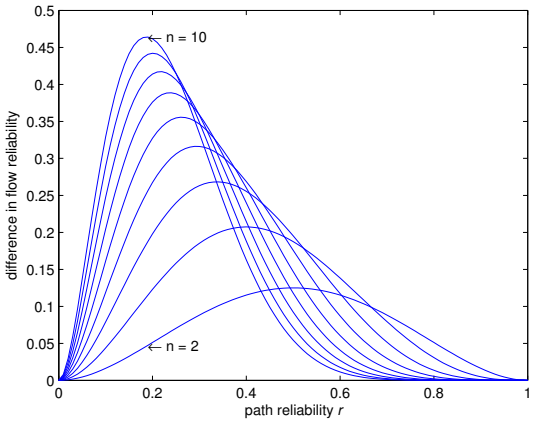


Figure 4.8: Difference in flow reliability of the SA and MA models, $r^{SA} - r^{MA}$.

To illustrate the difference between the two models in terms of flow reliability, I consider the case in which all sc- and ca-paths have the same reliability r. In this particular example, $r^{SA} - r^{MA} = [1-(1-r)^n]^2 - [1-(1-r^2)^n]$. I plot in a graph the difference (Fig. 4.8) varying $r \in [0,1]$ and $n \in \{2,...,10\}$. I observe that, as the number of paths $n \to \infty$, the maximum difference in reliability has upper bound 1. The big difference is obtained at small r.

2. Number of execution times of the control algorithm. For the SA model, each activation of a flow requires one execution of the control algorithm. For the MA model, it requires n executions of the control algorithm, where n is the number of sa-paths. Thus, SA is more favorable, especially for a computationally expensive control algorithm.

3. Flow latency. Given the same multi-path routing of a flow, the best-case flow latency of the SA model is equal to the sum hop count of the longest sc-path and the

longest ca-path, $\max_i\{\text{hops}(\pi_i^{sc})\} + \max_j\{\text{hops}(\pi_j^{ca})\}$, where $\text{hops}(\cdot)$ denotes the hop count of a path. On the other hand, the best-case flow latency of the MA model is equal to the hop count of the longest sa-path, $\max_i\{\text{hops}(\pi_i^{sc}) + \text{hops}(\pi_i^{ca})\}$. In this regard, the MA model is better. However, normally the routing problem is to find the lowest cost path satisfying a certain flow reliability level. Given the same routing, the SA model offers higher flow reliability. Furthermore, it allows finer granularity in path selection, meaning that when the flow reliability cannot be satisfied, instead of adding a new sa-path in the MA model, the SA model may add an sc-path or a ca-path. Therefore, the actual flow latency of the SA model may be even lower than that of the MA model.

Taking into account all of the above aspects, I prefer the SA scheduling model and use it consistently in this work.

4.2.4 *TDMA Scheduling Algorithms*

After I have established the system model, I now discuss the algorithms for scheduling TDMA transmissions. First I propose the framework for TDMA scheduling, viewing it as a multiprocessor scheduling problem. Then I describe a number of algorithms to be evaluated, and finally, I propose two concrete techniques: namely, *opportunistic aggregation* and *repetitive scheduling* which work seamlessly with any scheduling algorithm. As confirmed later by evaluation, the opportunistic aggregation significantly increases the schedulability while the repetitive scheduling significantly reduces the schedule table size and the execution time of the scheduling when the periods are harmonic and the deadlines are implicit.

4.2.4.1 *The Framework of Scheduling Algorithms*

Multiprocessor scheduling deals with the problem of assigning a set of tasks (sporadic or periodic) to a number of processors so that each task meets its deadline. My TDMA scheduling problem can be formulated as a multiprocessor scheduling problem where tasks can be of different granularities, such as the transmissions of a flow, of an sc- or ca-path, or of a link, and processors correspond to the available wireless channels. If there are N processors, the number of simultaneously scheduled tasks is at most N. Analogously, given N channels, the TDMA scheduling allows at most N transmissions in the same slot as the spatial reuse is disabled. However, my TDMA scheduling exhibits one peculiarity: any two scheduled links in the same slot cannot be in primary conflict, i.e., they cannot share a common node (with the exception of opportunistic aggregation). This implies that two tasks may be mutually exclusive, i.e., only one of them can be scheduled at a time.

There are two categories of multiprocessor scheduling algorithms: partitioned and global. The first one assigns a task statically to a processor, while the latter permits tasks to freely migrate between processors [DB11a]. Since migration corresponds to scheduling two consecutive transmissions on a path on two channels, and the cost of migration is negligible in my case, global scheduling should perform better. Thus I only investigate algorithms of this category.

Alg. 8 gives a high-level description of the general scheduling algorithm framework. Different algorithms follow the same steps, and they only differ from one another in

Algorithm 8: The scheduling algorithm framework.

```
Data: C: number of channels available
perform the schedulability test;
for slot = 0 to hyperperiod − 1 do
    collect all released transmissions;
    order them according to a fixed priority rule;
    from high to low priority, schedule at most C non-conflicting transmissions;
    if deadline is missed then
        return infeasible;
    if scheduling is complete then
        return feasible;
```

the way the *released transmissions* are prioritized where a released transmission refers to a transmission that can be scheduled in the next slot. First of all, a *schedulability test* is performed, and if it fails, i.e., a problem is definitely unschedulable, the algorithm stops. For the schedulability test, two checks are carried out:

1. Deadline Check. The deadline of a flow must be no smaller than the sum hop count of the longest sc-path and the longest ca-path.

2. Utilization Check (skipped for opportunistic aggregation). The utilization of a flow f is defined as $\frac{\text{hops}(f)}{\text{period}(f)}$, where $\text{hops}(f)$ is the number of transmissions for each activation of f and $\text{period}(f)$ is the flow period. The total utilization of all flows should not be larger than the number of channels. This is an obvious necessary schedulability test [Hor74].

If both tests are passed, it starts to perform scheduling. Although it is theoretically possible that a feedback loop has the deadline larger than the period, I impose that deadlines are less than or equal to periods, since: 1) if the periodic control system is feasible with the deadline larger than the period, a system would as well be feasible with a larger period but a restricted or implicit deadline (deadline $\leqslant$ period), which saves transmission cost; 2) according to [CG06], a synchronized (initially all tasks are released simultaneously), implicit- and restricted-deadline periodic taskset is schedulable, if it is feasible for a *hyper-period*, which is defined as the least common multiple of all task periods. Moreover, if deadlines were arbitrary, although the scheduling is still periodic after some point in time, this instant can not be accurately determined [CG07].

If the test outcome is positive, then the scheduling algorithm starts, and all released transmissions for the current slot are collected. However, there is one restriction: for one activation of a flow, the transmissions on the ca-flow are not released until all transmissions on the sc-flow are finished.

The Fig. 4.9 illustrates a scheduling problem with two channels. It serves as a running example for the explanation of the various scheduling algorithms. I have mentioned previously that algorithms differ in the way they prioritize tasks; the following subsection classifies and describes algorithms according to this criterion.

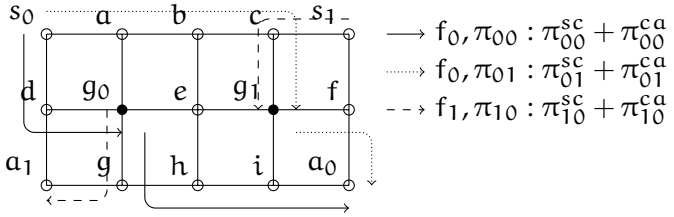


Figure 4.9: The example scheduling problem: a grid network of 15 nodes. g_0, g_1 are gateways. Two flows f_0 (sensor s_0, actuator a_0), f_1 (sensor s_1, actuator a_1) are to be scheduled. f_0 has two sa-paths ($\pi_{00} = \pi_{00}^{sc} + \pi_{00}^{ca}$ and $\pi_{01} = \pi_{01}^{sc} + \pi_{01}^{ca}$). f_1 has one sa-path ($\pi_{10} = \pi_{10}^{sc} + \pi_{10}^{ca}$) . The periods and deadlines of f_0 and f_1 are $p_0 = 10, d_0 = 10$ and $p_1 = 20, d_1 = 9$ respectively.

4.2.4.2 *Fixed Priority Scheduling Algorithms*

Fixed priority scheduling algorithms assign fixed priorities to tasks in the initialization phase, therefore no runtime information is required for priority comparison. Generally, the scheduling quality of *fixed priority scheduling* algorithms is worse than that of *dynamic priority scheduling* algorithms [DB11a], but they have the advantage of the possibility of schedulability analysis.

RATE MONOTONIC The Rate Monotonic (RM) algorithm is a classical algorithm [Liu00], which assigns higher priority to tasks of shorter period. I map the transmissions of a flow to a task. The flows of shorter period are assigned higher priorities. Fig. 4.10 shows that the transmissions of f_1 is delayed in slot 1 because of its lower priority.

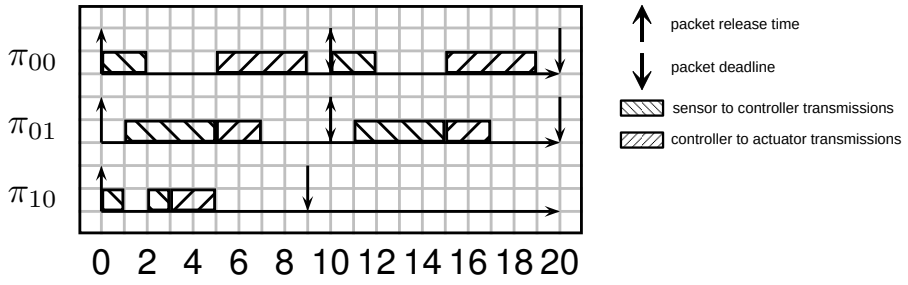


Figure 4.10: Scheduling with RM. The x-axis indicates slot ID.

DEADLINE MONOTONIC The Deadline Monotonic (DM) algorithm is another classical algorithm [Liu00], which assigns higher priority to tasks with shorter relative deadline (d_0, d_1 in my example). Same as the RM algorithm, I map the transmissions of a flow to a task. Therefore, f_1 is prioritized as shown in Fig. 4.11

PROPORTIONAL DEADLINE MONOTONIC The Proportional Deadline Monotonic (PDM) algorithm is proposed by [SXLC10]. I have adapted it to my system model. In comparison with RM and DM, the tasks have finer granularity — I map the

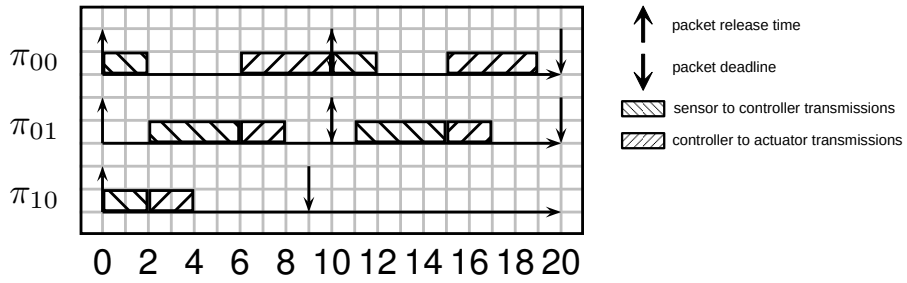


Figure 4.11: Scheduling with DM. The x-axis indicates slot ID.

transmissions of an sc- or ca-path to a task. The priority of each path is set according to the *proportional deadline*.

$$\mathrm{pd} = \frac{\mathrm{subflow\,deadline}}{\mathrm{path\,length}} \tag{4.17}$$

The smaller the value, the higher the priority. The *subflow deadline* of an sc- or ca-path is defined as the relative flow deadline minus the hop count of the longest ca- or sc-path, respectively.

Given the example problem, for flow f_0, the proportional deadlines of the four paths are:

$$\begin{aligned}
\mathrm{pd}(\pi_{00}^{sc}) &= \frac{d_0 - \max\{\mathrm{hops}(\pi_{00}^{ca}), \mathrm{hops}(\pi_{01}^{ca})\}}{\mathrm{hops}(\pi_{00}^{sc})} &= \frac{10 - \max\{4,2\}}{2} = 3 \\
\mathrm{pd}(\pi_{00}^{ca}) &= \frac{d_0 - \max\{\mathrm{hops}(\pi_{00}^{sc}), \mathrm{hops}(\pi_{01}^{sc})\}}{\mathrm{hops}(\pi_{00}^{ca})} &= \frac{10 - \max\{2,4\}}{4} = 1.5 \\
\mathrm{pd}(\pi_{01}^{sc}) &= \frac{d_0 - \max\{\mathrm{hops}(\pi_{00}^{ca}), \mathrm{hops}(\pi_{01}^{ca})\}}{\mathrm{hops}(\pi_{01}^{sc})} &= \frac{10 - \max\{4,2\}}{4} = 1.5 \\
\mathrm{pd}(\pi_{01}^{ca}) &= \frac{d_0 - \max\{\mathrm{hops}(\pi_{00}^{sc}), \mathrm{hops}(\pi_{01}^{sc})\}}{\mathrm{hops}(\pi_{01}^{ca})} &= \frac{10 - \max\{2,4\}}{2} = 3
\end{aligned}$$

For flow f_1, the proportional deadlines of the two paths are:

$$\begin{aligned}
\mathrm{pd}(\pi_{10}^{sc}) &= \frac{d_1 - \mathrm{hops}(\pi_{10}^{ca})}{\mathrm{hops}(\pi_{10}^{sc})} &= \frac{9-2}{2} = 3.5 \\
\mathrm{pd}(\pi_{10}^{ca}) &= \frac{d_1 - \mathrm{hops}(\pi_{10}^{sc})}{\mathrm{hops}(\pi_{10}^{ca})} &= \frac{9-2}{2} = 3.5
\end{aligned}$$

I can observe from Fig. 4.12, in the scheduling, π_{01}^{sc} has higher priority than π_{00}^{sc}. In addition, the paths of f_1 have lower priority than that of f_0.

4.2.4.3 *Dynamic Priority Scheduling Algorithms*

Dynamic priority scheduling algorithms determine the priority of a task at runtime, and they generally provide better scheduling quality than *fixed priority scheduling* algorithms.

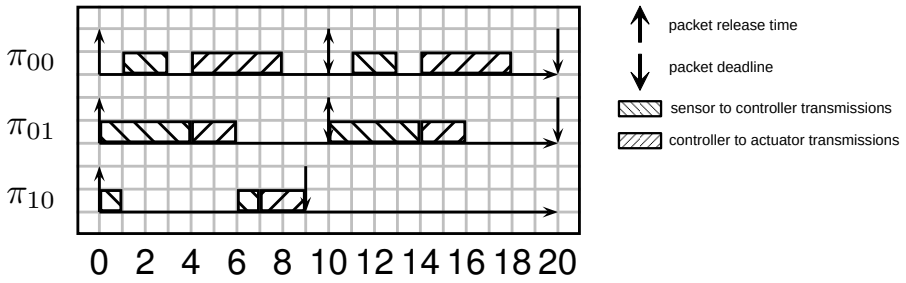


Figure 4.12: Scheduling with PDM. The x-axis indicates slot ID.

CONFLICT-AWARE LEAST LAXITY FIRST The Conflict-aware Least Laxity First (CLLF) algorithm is proposed by [SXLC10]. It is the least laxity first scheduling algorithm adapted to WirelessHART scheduling problem by taking into account the conflicts between pending transmissions. Because it needs to look into unreleased transmissions, it is computationally more expensive. The authors report that it performs significantly better than the other scheduling algorithms evaluated. Since I use a different system model, I have adapted the algorithm to my setting — by properly computing the deadline of a transmission. The scheduling is shown in Fig. 4.13.

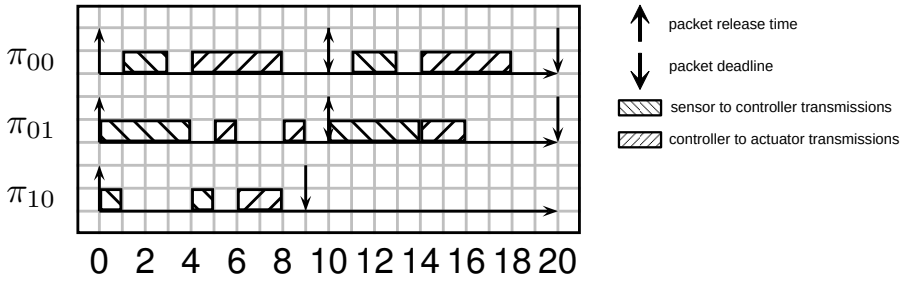


Figure 4.13: Scheduling with CLLF. The x-axis indicates slot ID.

As an example, I take a look at the scheduling of slot 4. The released transmissions are $\overrightarrow{g_0g}(\pi_{00})$, $\overrightarrow{g_1f}(\pi_{01})$ and $\overrightarrow{cg_1}(\pi_{10})$. The scheduling window for $\overrightarrow{g_0g}(\pi_{00})$ is $[4,6]$ which means the transmission must happen no earlier than slot 4 and no later than slot 6. Its conflicting transmissions set (the pending transmissions whose release time falls within the scheduling window and involving the sending node) has one element $\overrightarrow{g_0g}(\pi_{10})$ with scheduling window $[5,7]$. Therefore, the laxity of transmission $\overrightarrow{g_0g}(\pi_{00})$ is 2 (the smallest of the laxities of window $[4,6]$ and $[4,7]$. Both have laxity 2). The scheduling window for $\overrightarrow{g_1f}(\pi_{01})$ is $[4,8]$. It has only one conflicting transmission $\overrightarrow{cg_1}(\pi_{10})$ with window $[4,6]$. Its laxity is 2 (window $[4,6]$ and $[4,8]$ have laxities 2 and 3). The scheduling window for $\overrightarrow{cg_1}(\pi_{10})$ is $[4,6]$. It has no conflicting transmissions because no transmissions involving node c fall in the window. Thus the laxity is 2. To schedule transmissions, I prioritize those with smaller laxity. Since the laxities of the

three transmissions are the same, I select $\overrightarrow{g_0 g}(\pi_{00})$ and $\overrightarrow{cg_1}(\pi_{10})$ as they have smaller deadline ($6 < 8$).

EARLIEST DEADLINE FIRST The Earliest Deadline First (EDF) algorithm is a popular dynamic priority scheduling algorithm [Liu00]. It prioritizes the jobs (instances of tasks) with earlier absolute deadlines. I map the transmissions of an sc- or ca-path in one period as a job. Suppose flow f have sc-paths π_i^{sc} and ca-paths $\pi_j^{ca}(i, j = 0, 1, ...)$. And the relative flow deadline is d. The relative deadlines of an sc- and a ca-path are defined as follows:

$$\begin{aligned} deadline(\pi_i^{sc}) &= d - \max_j\{hops(\pi_j^{ca})\} \\ deadline(\pi_j^{ca}) &= d \end{aligned} \tag{4.18}$$

Then, I can compute the absolute deadline of the kth $(k = 0, 1, ...)$ activation of a path. Suppose p is the period. [6]

$$\begin{aligned} abs\_deadline(\pi_i^{sc}, k) &= k \cdot p + deadline(\pi_i^{sc}) - 1 \\ abs\_deadline(\pi_j^{ca}, k) &= k \cdot p + deadline(\pi_j^{ca}) - 1 \end{aligned} \tag{4.19}$$

Given the example problem, I have the absolute deadlines of all jobs in Tab. 4.2. The scheduling result of Fig. 4.14 shows that the priorities of various paths follow the absolute deadlines. For example, at slot 1, π_{10}^{sc} has lower priority than the other two active paths (π_{00}^{sc} and π_{01}^{sc}) because the absolute deadlines of the three paths are 5, 5 and 6, respectively.

path	π_{00}^{sc}		π_{00}^{ca}		π_{01}^{sc}		π_{01}^{ca}		π_{10}^{sc}	π_{10}^{ca}
period id	0	1	0	1	0	1	0	1	0	0
absolute deadline	5	15	9	19	5	15	9	19	6	8

Table 4.2: The absolute deadlines of the activations of the sc-/ca-paths.

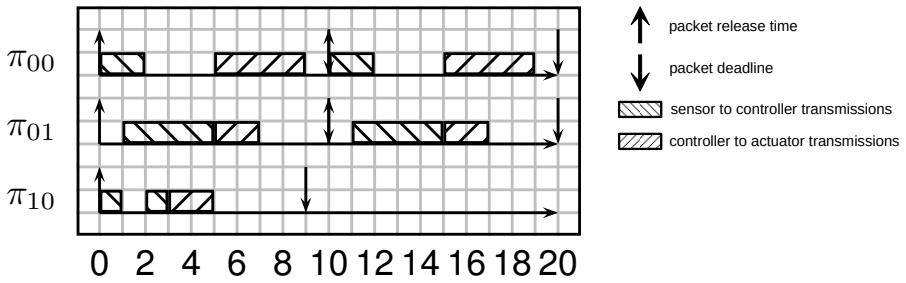


Figure 4.14: Scheduling with EDF. The x-axis indicates slot ID.

LEAST LAXITY FIRST The Least Laxity First (LLF) algorithm is another popular dynamic priority scheduling algorithm [Liu00]. The basic idea is to prioritize the

6 The -1 in Eq. 4.19 is due to that time slot id starts with 0.

jobs with smaller laxity left. For my problem, I compare the laxities of all released transmissions where the laxity is defined as the absolute deadline of the transmission minus the current time. For a released transmission of a given hop in an sc- or ca-path, its absolute deadline is computed as the absolute deadline of the path (Eq. 4.19) minus the remaining number of hops in the path. If two released transmissions have same laxity, I compare the *number of conflicting transmissions left* (*cfl_left*) which is defined as the sum of remaining number of transmissions on all conflicting links and itself till the end of the scheduling. The links with larger cfl_left are prioritized. The reason for doing so is that such links are probably bottlenecks and may harm parallelism of transmission as discussed in the *busy-sender-first* algorithm for tree-based convergecast (c.f. Sec. 4.1). [YH12b].

The scheduling on the example problem is shown in Fig. 4.15. Let us look at the scheduling in slot 3. At the moment, π^{sc}_{00} has finished transmissions, but π^{sc}_{01} has not. Therefore, there is no active transmission on the former path. The active transmission of π^{sc}_{01} is $\overrightarrow{cg_1}$, which has absolute deadline 5. The active transmission of π^{sc}_{10} is also $\overrightarrow{cg_1}$, but its absolute deadline is 6. The two transmissions have laxities 2 and 3 respectively. The *cfl_left* of both are 6 (the sum of remaining transmissions on $\overrightarrow{cg_1}$, $\overrightarrow{bc}$, $\overrightarrow{s_1c}$ and $\overrightarrow{g_1f}$). Because they are in conflict, only π^{sc}_{01} is scheduled.

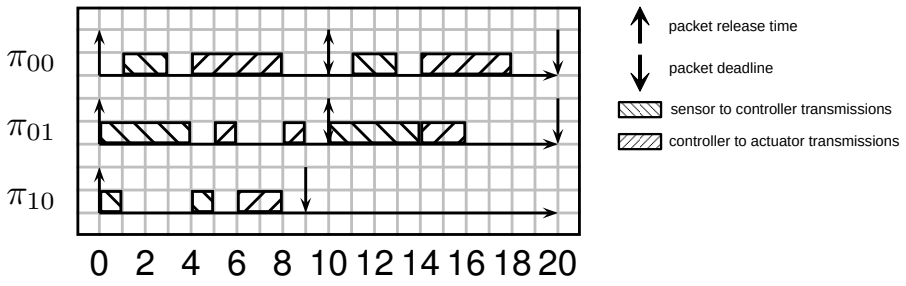


Figure 4.15: Scheduling with LLF. The x-axis indicates slot ID.

EARLIEST PROPORTIONAL DEADLINE The Earliest Proportional Deadline (EPD) algorithm is also introduced by [SXLC10]. It is a dynamic priority scheduling algorithm which prioritizes jobs with small sub-deadlines, where sub-deadline is defined as the time till the deadline of a job divided by the processing cost needed to finish the job. For my problem, the transmissions in an sc- or ca-path are mapped to a job. Sub-deadline is defined as the remaining slots till the path deadline divided by the remaining number of transmissions in the path. Suppose k is the current period id. s is current slot id.

$$\begin{aligned} sub\_deadline(\pi_i^{sc}) &= \frac{abs\_deadline(\pi_i^{sc}, k) - s + 1}{number\ of\ transmissions\ left\ in\ \pi_i^{sc}} \\ sub\_deadline(\pi_j^{ca}) &= \frac{abs\_deadline(\pi_j^{ca}, k) - s + 1}{number\ of\ transmissions\ left\ in\ \pi_j^{ca}} \end{aligned} \tag{4.20}$$

The scheduling is shown in Fig. 4.16. Taking slot 1 as an example, I have

$$\begin{aligned}
\text{sub_deadline}(\pi_{00}^{sc}) &= \frac{\text{abs_deadline}(\pi_{00}^{sc}, 0) - 1 + 1}{2} = \frac{5-1+1}{2} = 2.5 \\
\text{sub_deadline}(\pi_{01}^{sc}) &= \frac{\text{abs_deadline}(\pi_{01}^{sc}, 0) - 1 + 1}{3} = \frac{5-1+1}{3} = \frac{5}{3} \\
\text{sub_deadline}(\pi_{10}^{sc}) &= \frac{\text{abs_deadline}(\pi_{10}^{sc}, 0) - 1 + 1}{1} = \frac{6-1+1}{1} = 6
\end{aligned}$$

by referring to Tab. 4.2. Therefore, π_{00}^{sc} and π_{01}^{sc} are scheduled.

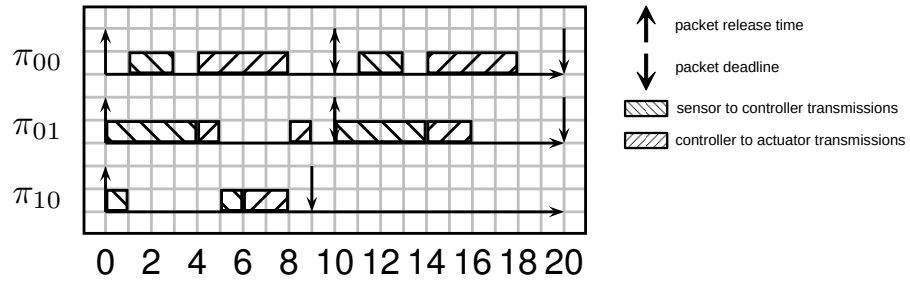


Figure 4.16: Scheduling with EPD. The x-axis indicates slot ID.

EARLIEST DEADLINE ZERO LAXITY The Earliest Deadline Zero Laxity (EDZL) algorithm is an effective dynamic priority scheduling algorithm proposed by Lee [Lee94]. It results in the same schedule as EDF until a situation is reached when a job will miss its deadline unless it executes for all of the remaining time up to its deadline (zero laxity). EDZL gives such a job the highest priority. For my problem, I implement EDZL as a mixture of EDF and LLF. If two jobs both have laxity greater than 0 and have unequal deadlines, EDZL works the same as EDF, otherwise, it works the same as LLF.

The scheduling is shown in Fig. 4.17. Let us take a look at the scheduling of slot 0. π_{00}^{sc} has deadline 5 (the latest time that transmissions in this path should finish) and laxity 4 (the released transmission $\overrightarrow{s_0 d}$ should not be scheduled later than 4, in order not to miss the deadline). π_{01}^{sc} has deadline 5 and laxity 2. π_{10}^{sc} has deadline 6 and laxity 4. Therefore, regarding priority, $\pi_{01}^{sc} > \pi_{00}^{sc} > \pi_{10}^{sc}$. Therefore, π_{01}^{sc} and π_{10}^{sc} are scheduled.

4.2.4.4 *Scheduling with Opportunistic Aggregation*

As explained above, a scheduling problem is definitely infeasible with C channels if the total utilization is greater than C. But packet aggregation can push the feasible total utilization over that limitation. Besides, a scheduling problem may be infeasible no matter how many channels are available, because the deadlines are too strict. As an example, consider 2 flows sharing a common path of $h (h \geqslant 2)$ hops. If the 2 flows are synchronized, i.e., packets are released at the same time, and the deadlines are h, the problem is obviously infeasible because it takes at least $h + 2$ slots to finish the transmissions. But, if packet aggregation is allowed, the deadlines of h can be met.

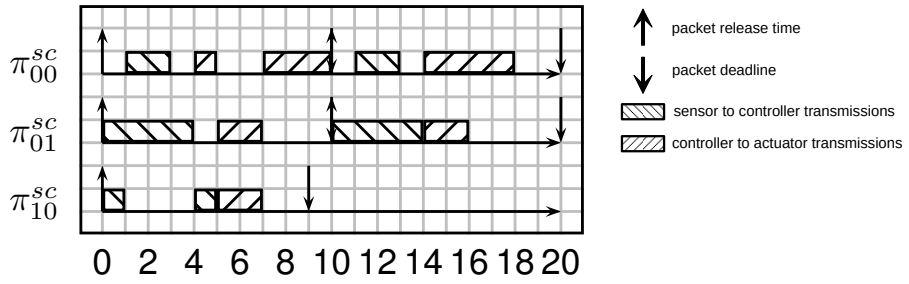


Figure 4.17: Scheduling with EDZL. The x-axis indicates slot ID.

Normally, TDMA protocols used for monitoring and controls (e.g., WirelessHART) have slot duration long enough for the transmission of a packet of the maximum size. And the packets of real-time control systems are generally very short, containing a few boolean, integer or floating point values, thus it is possible to aggregate a number of packets into one.

I design a scheme called *opportunistic aggregation* which works seamlessly with any scheduling algorithm. The feature of opportunistic aggregation is that, instead of scheduling according to priority (line 5 of Alg. 8), I execute Alg. 9 to determine which packets are to be scheduled and aggregated. The piece of code assumes the ideal case that unlimited number of packets can be aggregated. This gives the upper bound on the capability of opportunistic aggregation. In real implementation, I can easily add the restriction that the aggregated packet cannot be over a maximum size. Line 3 to 6 is the logic for packet aggregation — it is performed when the sender of a transmission is already scheduled, but the receiver is *free* (not scheduled) or the same link has been scheduled. Line 7 to 11 is the logic for scheduling a transmission on a link whose sender and receiver are both free. Another modification is in the schedulability test. I need to remove the utilization check as the total utilization may be larger than the number of channels available.

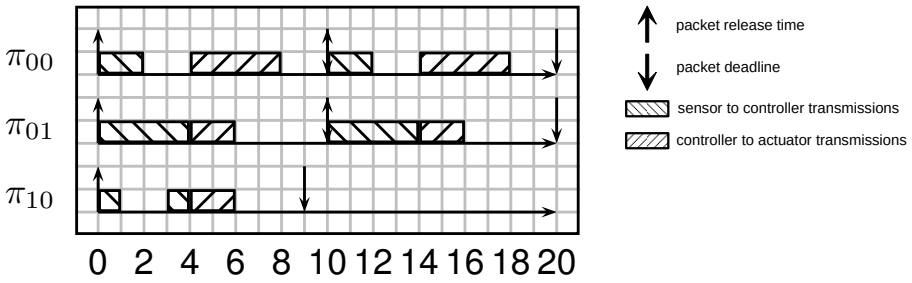


Figure 4.18: Scheduling with LLF plus opportunistic aggregation. The x-axis indicates slot ID.

Applying LLF with opportunistic aggregation to the example problem, I have the schedule in Fig. 4.18. I can see that opportunistic aggregation take effect in slot 0, 3, 4, 5 and 10. In slot 0, the same packet is sent on both links $\overrightarrow{s_0a}$ and $\overrightarrow{s_0b}$ simultaneously.

Algorithm 9: The logic of *opportunistic aggregation*.

```
Data: T : released transmissions ordered by priority, C : number of channels
1  m = 0;
2  for each transmission t in the sequence of T do
       // S: the set of scheduled senders, R: the set of scheduled
          receivers, L: the set of scheduled links
3      if t.sender ∈ S then
4          if (t.receiver ∉ S and t.receiver ∉ R) or (t.sender → t.receiver) ∈ L then
5              schedule t, do aggregation;
6              R = R + t.receiver, L = L + (t.sender → t.receiver);
7      else
8          if m < C and t.sender ∉ R and t.receiver ∉ S and t.receiver ∉ R then
9              schedule t;
10             S = S + t.sender, R = R + t.receiver, L = L + (t.sender → t.receiver);
11             m = m + 1;
```

In slot 3, two packets from different flows are aggregated and sent on link $\overrightarrow{cg_1}$. In slot 5, two packets from different flows are aggregated on node g and sent on links $\overrightarrow{ga_1}$ and $\overrightarrow{gh}$ simultaneously.

4.2.4.5 *Repetitive Scheduling for Systems with Harmonic Periods*

The TDMA scheduling of a periodic control system as described in this thesis is carried out at a central network manager and then sent to each mote. Each mote needs to store the part of the schedule table related to it. Therefore, I should not ignore the communication cost for downloading the schedule table and the memory cost for its storage.

Suppose the communication and storage cost of each table entry (one unaggregated packet transmission) is constant, then the sum cost of both are $\mathcal{O}(H \cdot U)$, where H is the hyper-period, and U is the total utilization. $H \cdot U$ is equal to the number of table entries. H is equal to the least common multiplier of all periods, $H = \text{lcm}(p_0, p_1, ..., p_{N-1})$, where N is the total number of flows. It can be as large as the products of all periods $\prod_i p_i$ when any two periods are co-prime. However, if the periods are *harmonic*, i.e., a period is divisible by any period that is smaller than it, then H is equal to the maximum period, $H = \max_i\{p_i\}$, for $i \in \{0, 1, ..., N-1\}$, leading to very small cost.

If a system has harmonic periods, I can further heavily reduce the communication and storage costs by scheduling it in such a manner that the schedule of each flow repeats in every period (like the schedule in Fig. 4.10). I call it *repetitive scheduling*. The other type that schedules every slot in a hyper-period is called *hyper-period scheduling*. The former has the total communication and storage costs $\mathcal{O}(\sum_i \text{hops}(f_i))$ where $\text{hops}(f_i)$ is the total number of hops in a flow f_i and it is independent of H. The latter has much larger costs, $\mathcal{O}(H \cdot U) = \mathcal{O}(\sum_i \frac{H}{p_i} \cdot \text{hops}(f_i))$. Therefore, the cost saving can be as large as $\frac{H}{\min\{p_i\}}$ fold. Another bonus is that the execution times of both schedulings are proportional to the storage costs, respectively. But there is a trade-off

— the repetitive scheduling has more restriction on the schedule, while may harm schedulability.

I give a method (Alg. 10) to do repetitive scheduling that works seamlessly with any scheduling algorithm. It schedules flows in the increasing order of period length. Because the periods are harmonic, if the scheduling of a flow meets the deadline for one period, it should always meet the deadline throughout the hyper-period. The method works independent of whether packet aggregation is applied or not.

Algorithm 10: The logic of *repetitive scheduling*. p_i's are harmonic.

Data: a hyper-period scheduling algorithm $\mathcal{A}$

1 collect all periods of N flows $p_0, p_1, ..., p_{N-1}$;

2 find distinct values from $p_i, i \in \{0, ..., N-1\}$, and order them as $p'_0 < p'_1 < ... < p'_{M-1}, M \leqslant N$;

3 **for** $i = 0$ **to** $M - 1$ **do**

4 schedule all flows of p'_i for one period using $\mathcal{A}$;

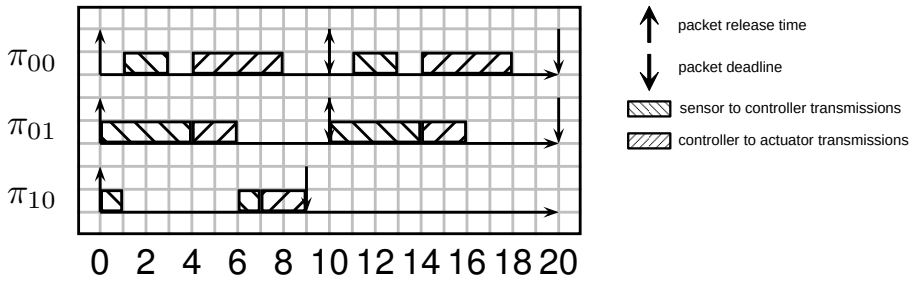


Figure 4.19: Repetitive scheduling with LLF (no aggregation). The x-axis indicates slot ID.

Applying repetitive scheduling of LLF on the example problem, I get the schedule in Fig. 4.19. Different from the scheduling in Fig. 4.15, the schedule for f_0 in two periods are exactly the same.

4.2.5 *Evaluation*

With randomly generated networks, I evaluate the performance of various scheduling algorithms, in order to identify the one with the best performance. Then I evaluate the effects of opportunistic aggregation and repetitive scheduling. The performance metrics of interest are *schedulability rate*, and practical ones such as execution time and memory consumption of an algorithm as well as schedule table size.

4.2.5.1 *The Evaluation Process*

In order to explore a large space of scheduling problems and observe the correlation of performance metrics with different variables, I vary a number of variables in the evaluation, including 1) different random topologies, 2) number of flows, 3) total

utilization, 4) implicit (deadline = period) or restricted deadline (deadline < period) [7] and 5) number of channels. The evaluation process is as follows:

TOPOLOGY GENERATION. Randomly generate a topology with 100 motes uniformly distributed within a square space (100 topologies are evaluated). Place 2 gateways to the centers of the left and the right half plane. Link qualities are determined according to the link quality model in Sec. 4.2.3.2. [8]

FLOW GENERATION. Generate a random number $F \in \{1, 2, ..., 50\}$ of flows (for each topology, I evaluate 5 different flow settings). [9]

ROUTING. Find two vertex-disjoint reliable paths for each flow. The sc-routing (from sensor to gateway) is done by first finding the most reliable path. Then remove all nodes on the path except the sensor (source), find again the most reliable path from the sensor to the other gateway. These two sc-paths are disjoint except at the sensor node. After that, I restore the original topology and find two disjoint ca-paths in the same way. Correspondingly, the two ca-paths are disjoint except at the actuator node (destination). Note an sc-path may share intermediate nodes with a ca-path. But the routing method has no single-point-of-failure in the sense that one node failure (an intermediate node or a gateway) will not cause the breakdown of the connection of a flow.

PERIOD AND DEADLINE SELECTION. Generate a random *expected total utilization* $\overline{U} \in (0, \overline{U}_{max})$ for the whole network, where $\overline{U}_{max} = 16$ when no packet aggregation is applied as this is the upper bound given 16 channels [10], otherwise $\overline{U}_{max} = 25$, which is the empirical upper bound for opportunistic aggregation. Alg. 11 is used to uniformly distribute $\overline{U}$ among all flows and set the period p_i of a flow f_i. p_i is chosen to be smallest value such that $u_i \leqslant \overline{u_i}$ where u_i and $\overline{u_i}$ are the actual utilization and the expected utilization of the flow, respectively (for each configuration of flows, I evaluate 10 different $\overline{U}$s). For the comparison between repetitive scheduling and hyper-period scheduling, a period p_i must be harmonic, i.e., $p_i = 2^k, k \in \{1, 2, ...\}$ and $p_i \leqslant 8192$. For other evaluations, p_i is a factor of 10000. Then choose an implicit or a restricted deadline. Alg. 11 makes use of the *UUniFast* taskset generation algorithm [BB05, DB11b], which can efficiently generate task sets with uniformly distributed utilization.

SCHEDULE WITH VARIOUS NUMBER OF CHANNELS. Given 1, 2, 4, 8 and 16 channels, schedule the problem. For the comparison of various algorithms, I run this step with all algorithms. Otherwise, I run it with the best scheduling algorithm LLF.

Note that, in real applications, normally the period of a flow is first specified by the control system requirements, then the routing is performed. As I want to control the total utilization, in order to investigate how it correlates with the scheduling performance, I reverse the two steps.

7 The deadline of a flow must not be less than the sum of the longest sc-path and ca-path.
8 Empirically, the generated networks have mean node degree of 5.5.
9 I require each flow to have two disjoint paths. This may not be feasible for all flows, therefore I may actually come up with less flows.
10 Remind that the IEEE 802.15.4 standard has 16 orthogonal channels in the 2.4 GHz band.

Algorithm 11: The algorithm for choosing flow periods.

Data: $\overline{U}$: expected total utilization, N: number of flows
Result: The period p_i for each flow f_i
// $u_i^{max} = \frac{hops(f_i)}{min\_period(f_i)}$ is the maximum utilization of a flow f_i, determined by the routing.
// $U^{max} = \sum_i u_i^{max}$ is the maximum total utilization.

```
1  sumU = min(Ū, U^max); discard = 0;
2  while true do
3      for i = 0 to N − 1 do
4          if i == N − 1 then
5              nextSumU = 0;
6          else
               // r is a random number, r ∈ (0, 1)
7              nextSumU = sumU · r^{1/(N−1−i)};
8          u_i = sumU − nextSumU;
9          if u_i > u_i^max then
10             discard++;
11             if discard == discard_limit then
12                 return generation failure
13             continue while;
14         sumU = nextSumU;
15     if harmonic periods then
16         p_i = min{p | p ⩾ ⌈hops(f_i)/u_i⌉, p = 2^k, k ∈ 1, .., 13}, ∀i;
17     else
18         p_i = min{p | p ⩾ ⌈hops(f_i)/u_i⌉, 10000 ≡ 0 (mod p)}, ∀i;
19     return generation success
```

	RM	DM	PDM	CLLF	EDF	LLF	EPD	EDZL
implicit	28.9	28.7	31.7	2917.0	26.2	52.6	29.2	27.8
restricted	8.8	9.1	9.4	288.5	8.4	14.5	8.8	8.3

Table 4.3: The mean execution time (in ms) of the 8 algorithms (hyper-period scheduling, w/o opportunistic aggregation).

4.2.5.2 *Hyper-period Scheduling without Packet Aggregation*

In this subsection, I evaluate the hyper-period scheduling without packet aggregation. My goal is to find the best scheduling algorithm and investigate the practical aspects of its performance such as the execution time and memory consumption.

COMPARISON OF VARIOUS ALGORITHMS Fig. 4.20 shows the cumulative schedulability rate (the percentage of the instances that are schedulable, i.e., all deadlines can be met) of the 8 algorithms given different number of channels. Tab. 4.3 shows the mean execution time of them.

LLF has the highest schedulability rate in almost all channel settings. EDZL also gives very good schedulability rate, being only slightly worse than LLF (the difference is within 1%). In comparison with the fastest algorithm, LLF takes about 1.7x to 2x the execution time. But in average it takes less than 53 ms, fast enough even for frequent on-line re-scheduling. Therefore, the LLF is the best algorithm among all, and in following evaluations, I will choose it implicitly.

Different from what is reported in [SXLC10], CLLF doesn't have better schedulability rate than the others [11], especially for the *restricted deadline*. In addition, it has the problem that giving more channels may sometimes decrease the schedulability rate (compare the results of 8 and 16 channels). Another drawback is that it is much slower, takes about 20x to 55x more time than LLF. Therefore, CLLF is not desirable.

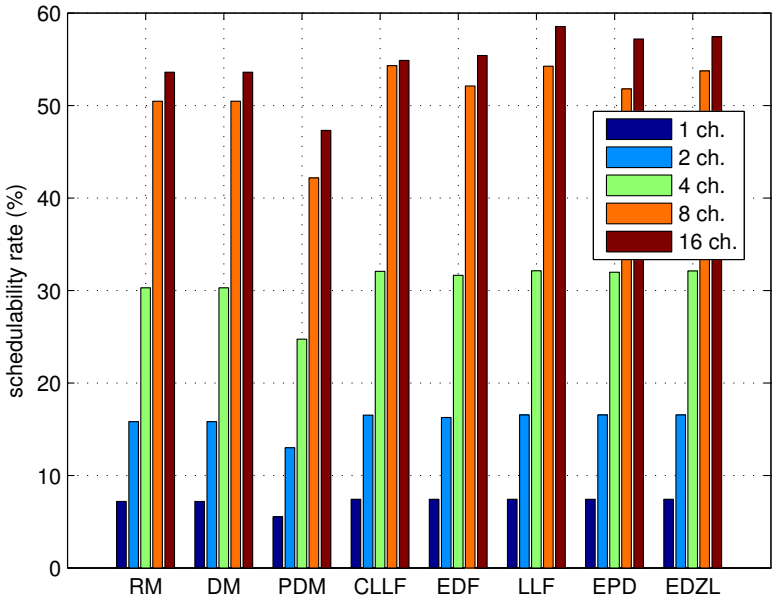
The fixed priority scheduling algorithms (RM, DM and PDM) perform slightly worse than the dynamic priority scheduling algorithms for the implicit deadline. However, for the restricted deadline, I should avoid fixed priority scheduling algorithms because their performance is significantly worse.

THE CORRELATION BETWEEN SCHEDULABILITY RATE WITH NUMBER OF CHANNELS AND TOTAL UTILIZATION Fig. 4.21 displays how schedulability rate changes with the number of channels and the total utilization. It shows that

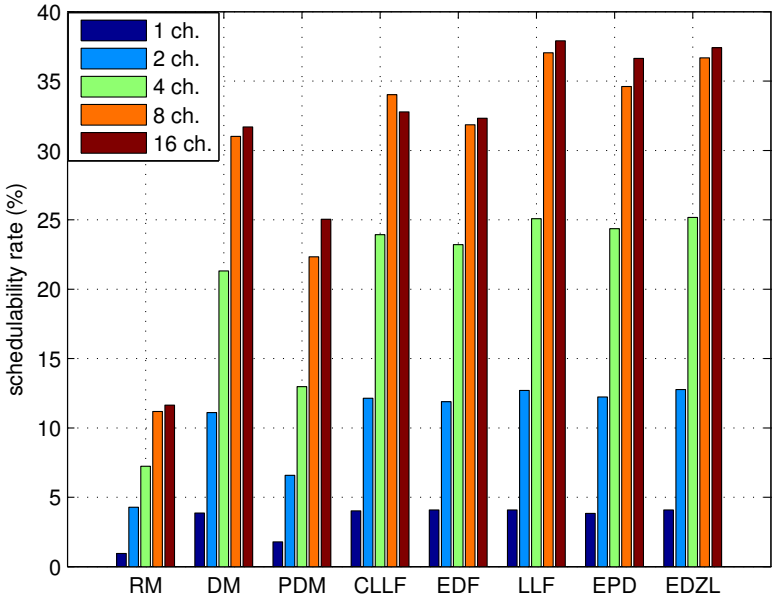
1. Given a total utilization value, the schedulability rate increases with the number of channels. However, the marginal improvement decreases (also shown by Fig. 4.20). My explanation is that some scheduling problems, are infeasible no matter how many channels are assigned. To make such problems schedulable, I should reduce primary conflict (e.g., by using packet aggregation) and shorten routing paths.

2. The schedulability rate decreases monotonically with the total utilization for a fixed number of channels. This matches the intuition more busy networks are

11 Note, my system model is a bit different from theirs. I use the SA model, but their work uses the MA model (c.f. Sec. 4.2.3.4).

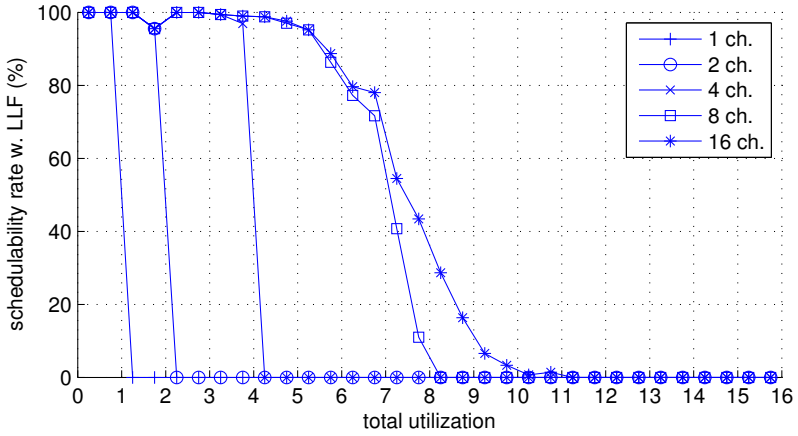


(a) *implicit deadline*

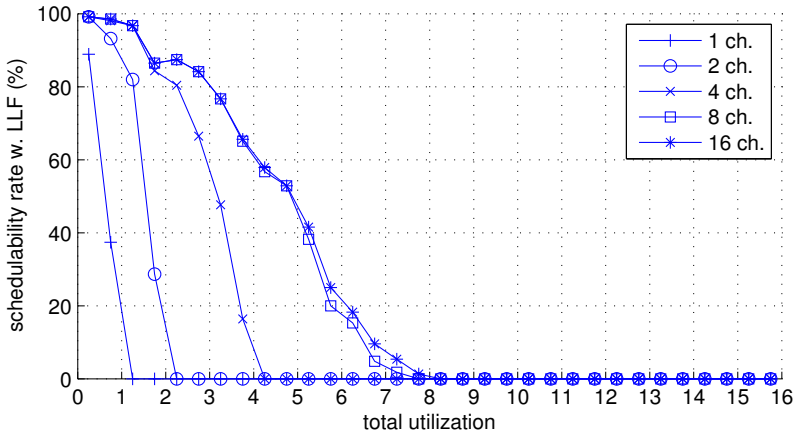


(b) *restricted deadline*

Figure 4.20: Schedulability rate of the 8 algorithms (hyper-period scheduling, w/o opportunistic aggregation).



(a) *implicit deadline*



(b) *restricted deadline*

Figure 4.21: How schedulability rate changes with the number of channels and total utilization (hyper-period scheduling, w/o opportunistic aggregation). A point (x, y) on the curves corresponds to the schedulability rate (y) of all scheduling problems which have its total utilization falling in the range of $(x - 0.25, x + 0.25]$ and have a certain number of channels. The separation of two neighboring points on a curve is 0.5 in x-axis.

less schedulable. Given the same total utilization, the restricted deadline case is significantly less schedulable than the implicit deadline case. And it has a wider transitional region of utilization in which the schedulability rate change from 100% to 0%. In addition, the plots match the theory that schedulable problems have the total utilization less than or equal to the number of channels.

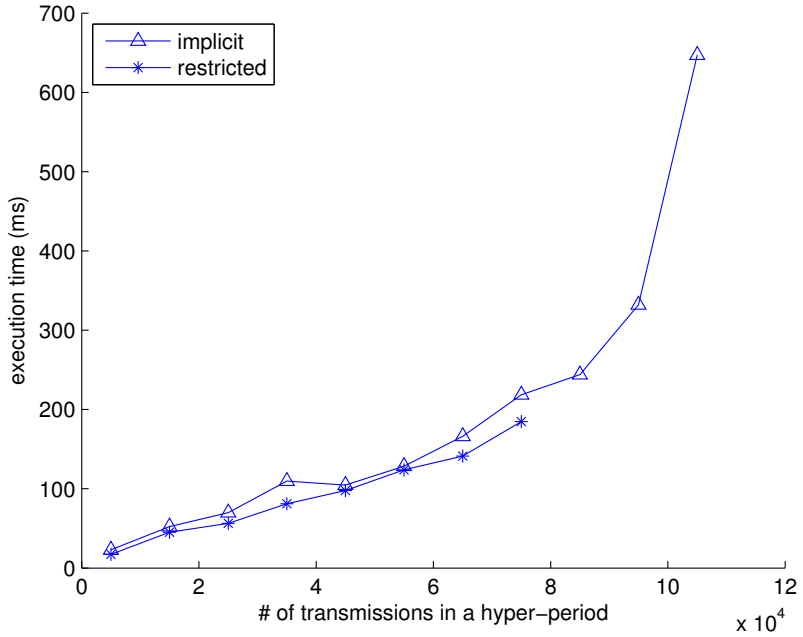


Figure 4.22: Execution time (hyper-period scheduling, w/o opportunistic aggregation). A point (x, y) on the curves corresponds to the mean execution time (y) of all schedulable problems with number of transmissions in $(x - 0.5 \cdot 10^4, x + 0.5 \cdot 10^4], x = \{0.5, 1.5, ..., 10.5\} \cdot 10^4$.

EXECUTION TIME Fig. 4.22 shows that execution time generally increases with the number of transmissions in a hyper-period, with a speed faster than linear. In addition, for the same number of total transmissions, implicit deadline requires a relatively larger execution time. From the dataset, I find the extreme value is 1.4 sec, still acceptable for frequent on-line rescheduling.

THE WORST-CASE BUFFER CONSUMPTION OF NETWORK OPERATION Because wireless sensor nodes normally have very limited memory resource, I need to investigate the memory consumption on the motes during the network operation. The metric of interest is the *maximum buffer consumption*, which is measured in number of packets and is defined as the maximum number of packets buffered at a mote during the network operation. I do not consider the gateways, as they can be wired and mains-powered, therefore their memory resource is not so limited.

From the scheduling results, I find that the *maximum buffer consumption* generally goes up with the number of flows and the total utilization linearly. Furthermore, given the same scheduling problem, the maximum buffer consumption also goes up with the number of channels linearly. This is explainable by the reasoning that more flows and more transmissions per timeslot lead to more competition for the same links,

therefore, packets are more likely to be buffered at motes. More channels allow more parallelism in transmissions which leads to more dynamical behavior of the memory consumption.

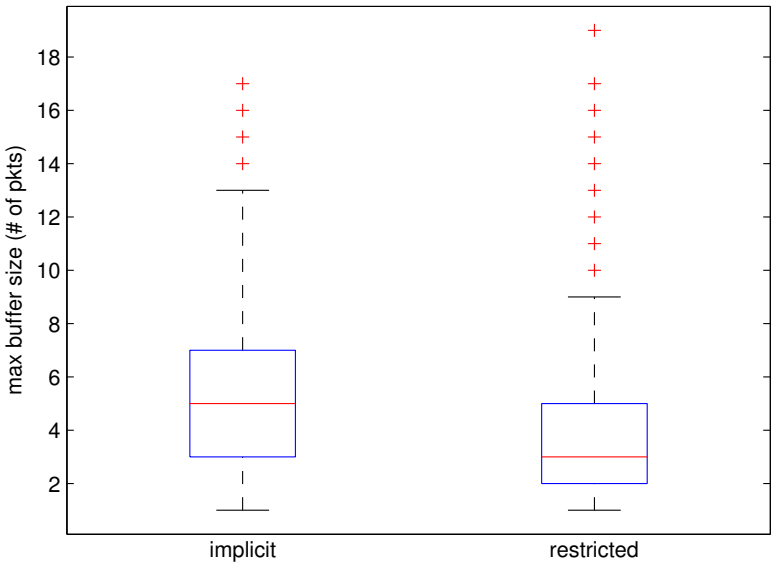


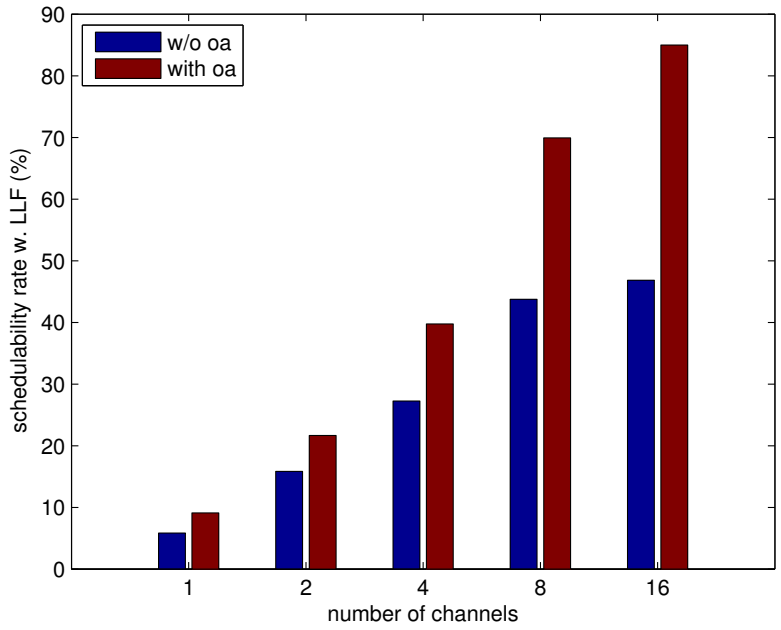
Figure 4.23: The box plots of *maximum buffer consumption* (hyper-period scheduling, w/o opportunistic aggregation). The central line is the median.

Recall that the evaluated networks have 100 motes, and up to 50 flows. The scale can be viewed as an upper bound for the industrial automation applications. From the box plots in Fig. 4.23, I can see the median *maximum buffer consumption* is less than 6. In the worst-case, 19 packets are buffered. Since a packet is no more than 127 bytes, this takes less than 2.5K bytes which can fit into the 10K bytes RAM of the most widely used TelosB mote [Moto4].

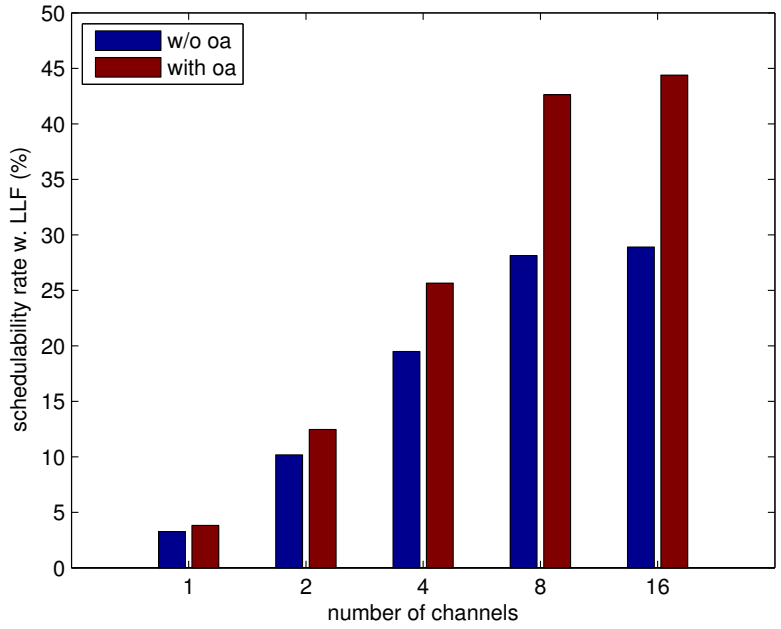
4.2.5.3 *Hyper-period Scheduling with Opportunistic Aggregation*

Now I evaluate how opportunistic aggregation affects scheduling.

SCHEDULABILITY RATE Fig. 4.24 compares the cumulative schedulability rate for the modes with opportunistic aggregation and without. Fig. 4.25 shows how the schedulability rate changes with the *total utilization* when *opportunistic aggregation* is applied, which should be viewed in comparison with Fig. 4.21. From the figures, I observe that for both deadline setups, opportunistic aggregation significantly improves the schedulability rate by 17% to 81%. The relative improvement for the implicit deadline is larger than that of restricted deadline. This is probably because the implicit deadline is less restrictive, giving opportunistic aggregation more room for improvement. Opportunistic aggregation is very effective in combating the primary conflicts, by resolving the competition for a sender to a large extent. The *aggregation rate*, i.e., the percentage of the packets that are aggregated among all packets, of the



(a) *implicit deadline*

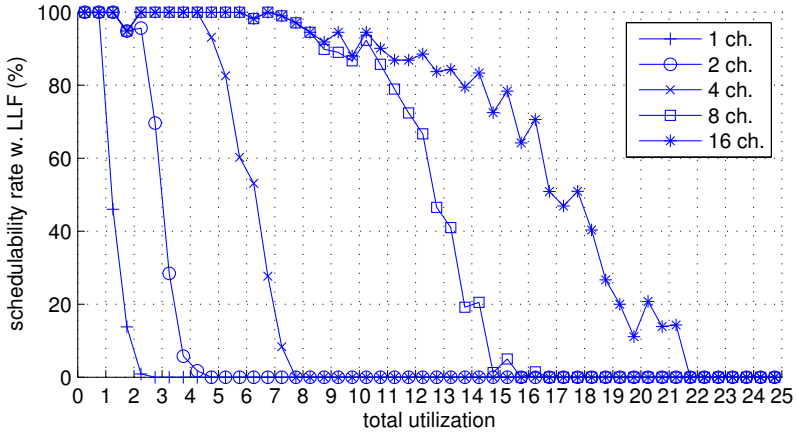


(b) *restricted deadline*

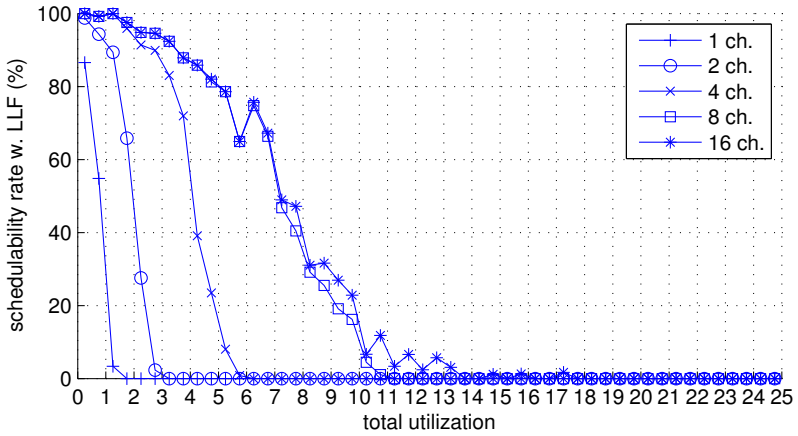
Figure 4.24: Schedulability rate (hyper-period scheduling, with opportunistic aggregation vs. w/o).

cases of the implicit/restricted deadline are 22% and 17% respectively. Surprisingly, with a moderate aggregation of packets, I increase the schedulability rate significantly.

From Fig. 4.25, I see that opportunistic aggregation really pushes the maximum schedulable total utilization to a value much higher than the number of channels C,



(a) *implicit deadline*



(b) *restricted deadline*

Figure 4.25: How schedulability rate changes with number of channels and total utilization (hyper-period scheduling, with opportunistic aggregation).

especially for the case of implicit deadline. It breaks the otherwise valid constraint $\mathtt{total utilization} \leqslant C$. If there is no packet aggregation, the schedulability rate of 16 channels only slightly improves that of 8 channels (Fig. 4.21). On the contrary, opportunistic aggregation gives large improvement from 8 to 16 channels for the case of implicit deadline, which much better exploits the precious channel resource.

EXECUTION TIME I find that with opportunistic aggregation, the scheduling in average takes 14% to 29% less time than without. This is counter-intuitive, because in the former case, to schedule a slot, I need to check every released transmission, while for the latter case, when all scheduling chances (equal to the number of channels) are used up, I don't need to check further. The shorter execution time is however due to the higher throughput for the mode of opportunistic aggregation. The released transmissions finish earlier and therefore fewer timeslots need to be scheduled.

Fig. 4.26 shows the change of the mean execution time with the number of transmissions in a hyper-period. Same as the mode without opportunistic aggregation, the

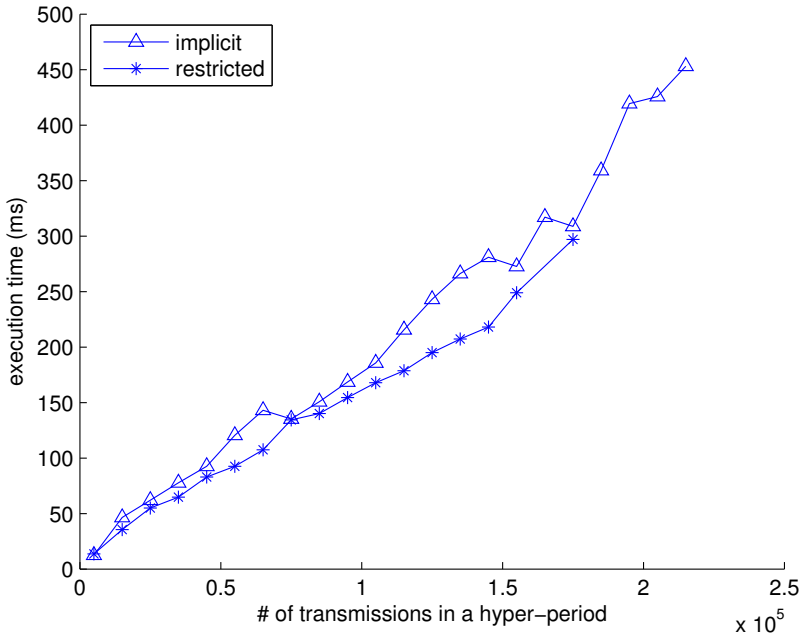


Figure 4.26: Execution time (hyper-period scheduling, with opportunistic aggregation).

execution time increases with the number of transmissions, but with an almost linear speed.

THE WORST-CASE BUFFER CONSUMPTION OF NETWORK OPERATION Same as the case without opportunistic aggregation, the maximum buffer consumption generally goes up with the number of flows and the total utilization linearly. Given the same scheduling problem, the maximum buffer consumption also goes up with the number of channel linearly.

Fig. 4.27 gives the box plots for the maximum buffer consumption. The median value is less than 7. In the worst case 27 packets are buffered, small enough ($<$ 3.5K bytes) to fit into the RAM of a TelosB mote. Therefore, in trading for the significant improvement in schedulability rate and execution time, the penalty on the buffer consumption is minimal.

4.2.5.4 *Comparison of Repetitive Scheduling and Hyper-period Scheduling*

Now I evaluate the advantage of the low overhead (in schedule table size and execution time) of the repetitive scheduling and the trade-offs it incurs in comparison with the hyper-period scheduling.

SCHEDULABILITY RATE Fig. 4.28 compares the schedulability rates of the repetitive scheduling and the hyper-period scheduling. It demonstrates that for the implicit deadline, repetitive scheduling decreases the schedulability rate only slightly in comparison with the hyper-period scheduling. Especially when no opportunistic aggregation is applied, both have almost the same schedulability rate. Therefore, the slight penalty is well worth the large cost reduction in execution time, communication and storage, which will be shown later. But for the restricted deadline, the repetitive

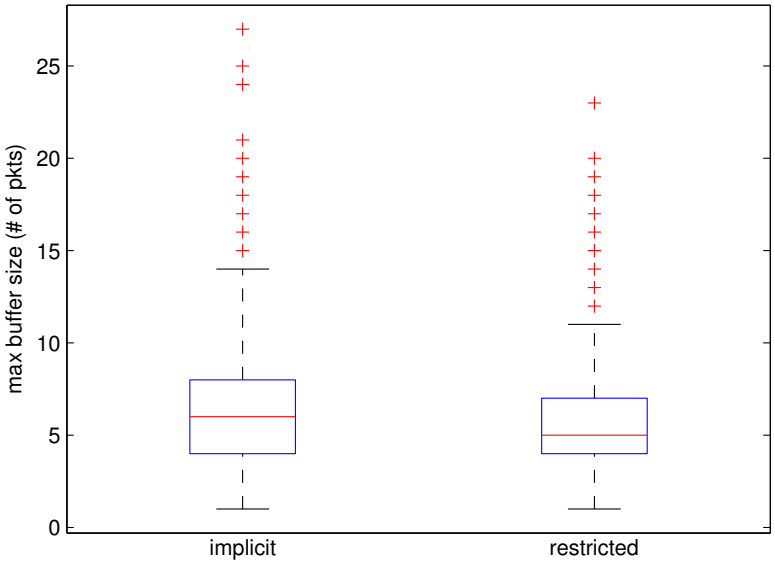


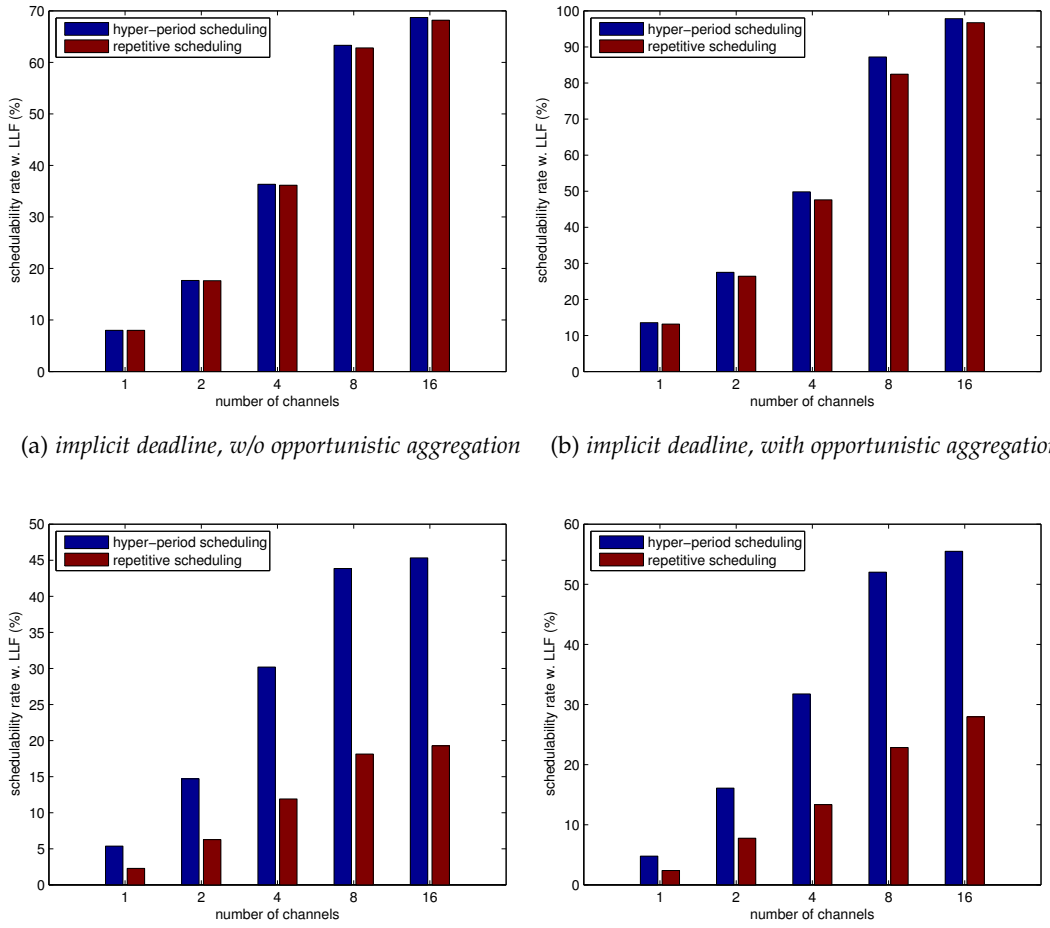
Figure 4.27: The box plots of *maximum buffer consumption* (hyper-period scheduling, with opportunistic aggregation).

scheduling causes a large decrease in schedulability rate, over 50%. Fortunately, most control systems have implicit deadline requirements, making repetitive scheduling very attractive.

LOW OVERHEAD OF THE REPETITIVE SCHEDULING Fig. 4.29 compares the number of entries in the respective schedule tables of the repetitive scheduling and the hyper-period scheduling for the same scheduling problems. Because the communication and storage cost is proportional to the number of entries, I see tremendous cost reduction under the repetitive scheduling. A good property of repetitive scheduling is the scalability of the cost — it almost remains constant while the cost of the hyper-period scheduling increases linearly. In the evaluated problems, the maximum number of table entries are 1330 and 147,811 for the two scheduling schemes respectively. Each entry may contain the following fields: slot id (2 bytes), channel id (4 bits), sender (1 byte), receiver (1 byte) and flow id (6 bits). In total, it takes no more than 5 bytes per entry. Thus, the maximum schedule tables would be 53.2 and 5,912.4 kbits respectively. With a convergecast throughput of 203kbit/s as achieved by [ZÖS+10], it is feasible to frequently refresh the schedule table with the repetitive scheduling while it takes much longer (100x) with the hyper-period scheduling if not impossible.

Fig. 4.30 shows another great advantage of repetitive scheduling — short execution time. The plots show the mean execution time of the scheduling versus the total number of transmissions in a hyper-period. I see that with the increase of the number of transmissions, the execution time increases linearly under the hyper-period scheduling while it almost stays constant (very scalable) under repetitive scheduling. [12]

12 For the restricted deadline, I observe similar results of table size and execution time.



(a) *implicit deadline, w/o opportunistic aggregation* (b) *implicit deadline, with opportunistic aggregation*

(c) *restricted deadline, w/o opportunistic aggregation* (d) *restricted deadline, with opportunistic aggregation*

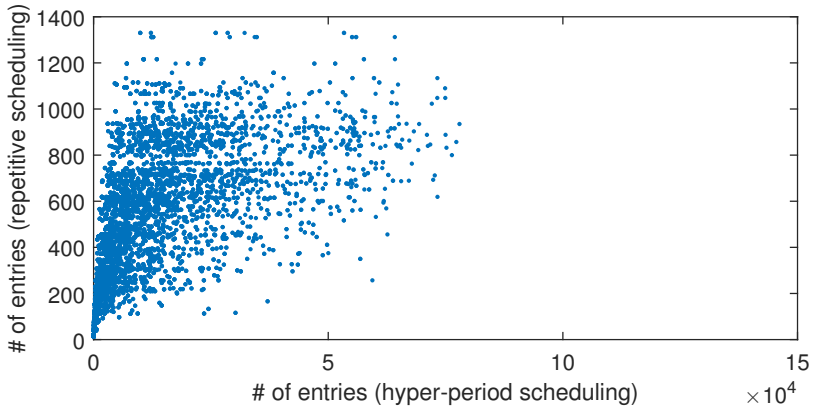
Figure 4.28: Schedulability rate (repetitive scheduling vs. hyper-period scheduling).

4.2.6 *Conclusion*

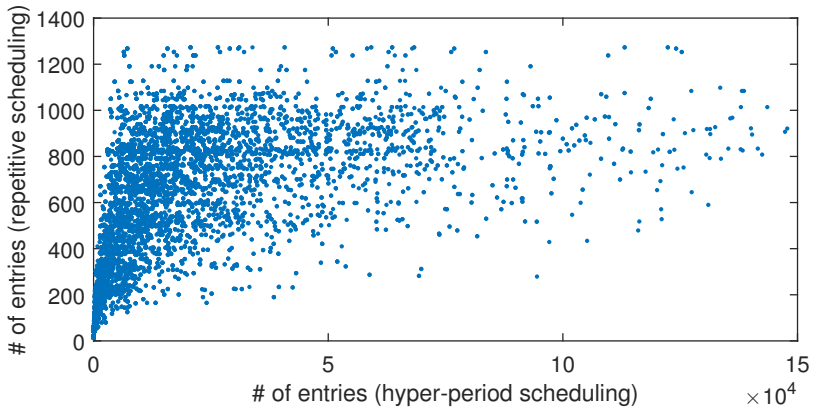
I have investigated the centralized TDMA scheduling of wireless sensor networks for multi-rate periodic control systems assuming a very general system model. Through extensive evaluation, I have identified LLF as the best scheduling algorithm with regard to high schedulability rate, low execution time and low memory overhead. I have proposed two new scheduling schemes — opportunistic aggregation and repetitive scheduling that work seamlessly with any scheduling algorithm and are easy to implement. The former significantly increases the schedulability rate while the latter incurs very low and scalable schedule table size and execution time. I demonstrate that LLF in combination with them, provides a practical solution to on-line centralized TDMA scheduling for WSN-based periodic control systems.

4.3 SUMMARY

Tree-based convergecast is a universal communication paradigm. It can be widely used as the underlying communication mechanism for single-rate industrial monitoring



(a) *implicit deadline, w/o opportunistic aggregation*

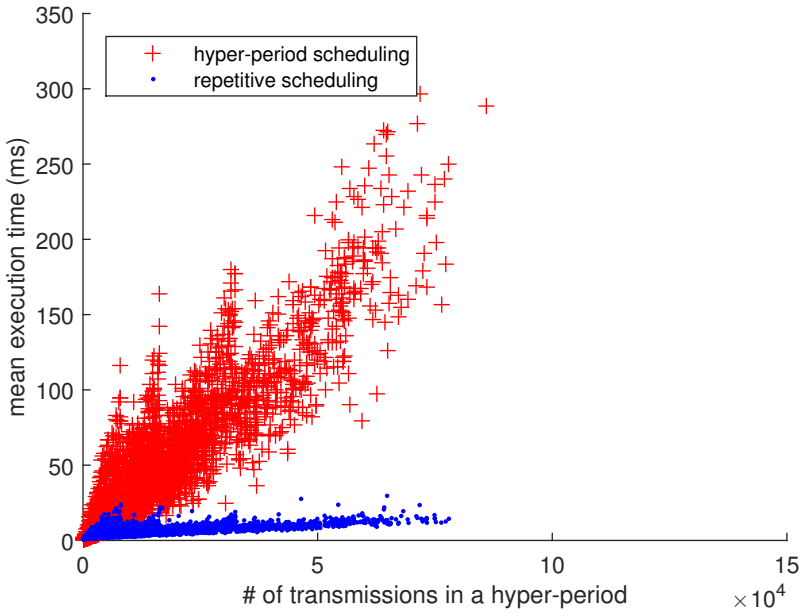


(b) *implicit deadline, with opportunistic aggregation*

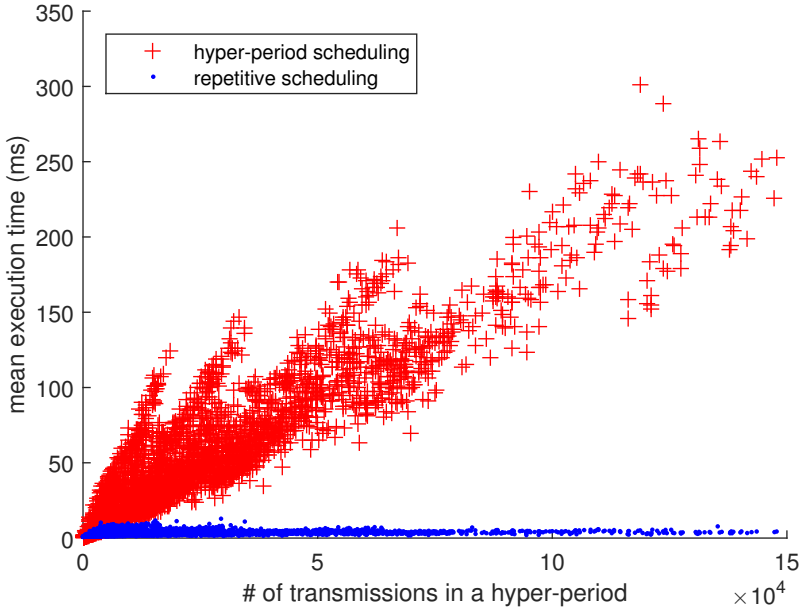
Figure 4.29: Number of entries in the schedule table (repetitive scheduling vs. hyper-period scheduling).

and control systems. Since the optimal solution for the minimum length scheduling is not yet available, effective heuristics are indispensable. The thesis proposed the very effective busy-sender-first heuristic, which is within 1.22% of the minimum schedule length. Compared to the state-of-the-art heuristic [ZSJ09a], it has significantly shorter schedule length, is more simple in concept and in implementation, and incurs slightly less memory consumption. I also give guidelines on choosing good topologies for tree convergecast. By also taking into account a reliable tree topology, I can find an optimal solution for the tree-based convergecast problem.

For multi-rate industrial monitoring and control systems, the TDMA scheduling in the second part of the chapter can be applied. Since the problem is shown to be NP-hard, effective heuristics are unavoidable. I discovered a very good scheduling heuristic LLF which achieves the highest schedulability rate among all heuristics evaluated. I also proposed two new scheduling schemes: opportunistic aggregation and repetitive scheduling which work seamlessly with any heuristic. The first scheme significantly increase the schedulability rate and the second incurs very low and scalable communication, storage and computation overhead.



(a) *implicit deadline, w/o opportunistic aggregation*



(b) *implicit deadline, with opportunistic aggregation*

Figure 4.30: Execution time (repetitive scheduling vs. hyper-period scheduling).

In all, my contributions in this chapter make TDMA scheduling more practical for industrial WSNs.

5

IMPROVING QUALITY OF SERVICE FOR INDUSTRIAL AUTOMATION WITH CONCURRENT TRANSMISSION

This chapter deals with modeling concurrent transmission [YH13] in wireless sensor networks and improving quality of service (QoS) of communication with concurrent transmission for industrial automation. The chapter is composed of three parts. The first part provides a comprehensive understanding of concurrent transmission in WSN. I first give a measurement-based model of one-hop delay of the Glossy implementation of concurrent transmission on widely used Telosb nodes. Based on that, I propose an accurate reception prediction model for concurrent transmission, which serves as a complete toolset for predicting concurrent transmissions for a single hop and a multi-hop network. The second part introduces the concurrent transmission technique to periodic multi-rate monitoring and control systems. I propose the protocol Sparkle [YRH14], which achieves high packet reliability, high energy efficiency as well as near-optimal and deterministic latency. Sparkle has a flexible and extensible architecture that supports arbitrary and independent QoS control mechanisms for all communication flows. I show that by adaptively selecting the transmission power and a suitable level of WSNShape — a topology control technique based on the capture effect, I can satisfy the design goal of a preset reliability while massively reducing energy consumption. The last part of the chapter improves on the concurrent transmission based network flooding protocol Glossy. The proposed Ripple protocol [YH15] can be ideally applied for bulk data dissemination such as broadcasting TDMA schedules, code or multimedia data. Ripple adds to Glossy pipeline transmission and error coding. The pipeline transmission on multiple channels employs a novel packet-based channel assignment and raises the throughput and energy efficiency multifold. The error coding pushes the packet reliability over that of Glossy, close to 100%. In addition, by tuning the transmission interval, Ripple can balance between throughput and reliability, suiting a large spectrum of applications with various QoS requirements.

5.1 AN EFFECTIVE MODEL FOR CONCURRENT TRANSMISSION

5.1.1 *Introduction*

As discussed in Chapter 2 *Background and Related Work*, WSNs are increasingly deployed in commercial and industrial applications with stringent requirements on communication latency and reliability. At the same time, the network nodes are expected to last for months or years without maintenance. Optimization of energy consumption is related to all parts of a WSN. Of particular importance are wireless communication techniques that operate with as low power as possible and advanced duty cycle algorithms. However, saving energy at all cost can lead to low throughput, high packet latency and low packet reliability: the exact opposite of what the above applications require.

Glossy [FZTS11] provides a promising technique to deal with the dilemma by exploiting Constructive Interference (CI), i.e., a number of nodes transmit the same packet at roughly the same time so that the signals add up constructively at the receiver. The spatial redundancy in combination with CI offers very high reliability and almost optimal latency. However, to obtain CI, an extremely tight synchronization among concurrent transmissions is a must. How well this can be achieved in real-world implementation is unclear. When the very tight synchronization is not met, capture effect still occurs with high probability, thus may still enable successful concurrent transmission. Yet, it is also unclear how the capture effect behaves in IEEE 802.15.4 WSNs. More general, we lack a complete view on the nature of concurrent transmission: this is the main contribution of Sec. 5.1.

My contribution is three-fold:

1. I experimentally examine the implementation of CI. In particular, I derive an accurate statistical model for the one-hop delay. I find that when the relative frequency skew of the radio oscillators on different concurrent transmitters is large and the packet length is long, it is hard to obtain CI even for the single hop setting.

2. I show by means of experiment under which conditions the capture effect takes place. I find that capture happens when a strong signal arrives no later than one preamble duration after the weak signal to which the receiver is synchronized. I identify a transitional region of three bytes in which capture can be observed to different degrees. No capture effect can be observed after this transitional region.

3. By jointly considering CI, capture effect and Signal to Interference plus Noise Ratio (SINR)/Symbol Error Rate (SER), I propose an accurate model for the prediction of the success of packet reception during concurrent transmission. I validate my model by experiment with up to six concurrent transmitters and show its high prediction quality.

5.1.2 *Related Work*

The first step to better exploit concurrent transmission in WSNs is to understand and model it. A seminal work studying concurrent transmission in WSN empirically is by Son et al [SKH06]. The paper describes a series of experiments carried out with the CC1000 radio [CC107] and their results proved that the SINR determines Packet Reception Rate (PRR) in concurrent transmission and that the capture effect is significant. Despite the relevance of the outcome of these tests, the influence of packet arrival timing on the capture effect was not addressed in this work, which is investigated in this thesis. My concurrent transmission prediction model builds on the SINR/PRR relationship, whose accuracy is confirmed in [MJD08a] for different number of interferers, multiple channels, and different transmission powers. Moreover, my experiments have been performed with the broadly used TelosB motes, equipped with an IEEE 802.15.4 compatible CC2420 radio chip [CC213]. As CC1000 radio employs a different modulation scheme, Frequency Shift Keying (FSK), and has lower data rate ($\leqslant$ 76.8 kbit/s) than the IEEE 802.15.4 standard, some of the conclusions in

[SKH06] may not be applied to the IEEE 802.15.4 standard. For example their claim about the non-additivity of the interference of concurrent transmissions is refuted by [MJD08b] for the CC2420 radio.

To my knowledge, [DMEST08] is the first work that identifies constructive interference (CI) by showing the superposition of up to a dozen of highly synchronized hardware Acknowledgement (ACK) packets can be decoded correctly. My study of CI draws primarily on the Glossy [FZTS11] open source implementation. This is the seminal work which laid the foundation of designing and implementing CI in WSN. I unambiguously show that CI boosts packet reliability by carefully placing four nodes so that their signal strengths are roughly the same at the receiver. In this way, I keep SINR below $-3dB$ for their concurrent transmission despite capture effect. With such low SINR I expect 0% PRR, but the actual PRR is 94%. I carefully model the one-hop delay experimentally by dividing it into 3 components: transmission duration, data latency and software delay. However, software delay exhibits a larger variance in my case than reported in [FZTS11]. My concurrent transmission prediction model makes use of the theoretical waveform analysis for CI, presented in [WHM+12].

Capture effect in WSN is studied in [WWJ+05, LW09]. They analyze the phenomena of strong-first and strong-last capture in the CC1000 and the CC2420 radios. I go one step further and show that the capture effect happens for certain when the strong signal arrives no later than preamble duration after the weak signal. Furthermore, it extends for another 3 bytes to different degree, after which no capture effect occurs.

My prediction model employs an SINR/BER (Bit Error Rate) model given by the IEEE 802.15.4 standard [Soc06], which assumes the interfering signal to be similar to the additive white Gaussian noise. It is suggested in [ZHZ+11] that the SINR/BER model is actually dependent on channel model.

5.1.3 *Constructive Interference*

When using the CI technique, it is of utmost importance to respect time synchronization in the transmission of concurrent identical packets. According to [FZTS11, WHM+12], time displacement between multiple packets should be no longer than 0.5µs, half of the Direct Sequence Spread Spectrum (DSSS) chip duration as specified in the IEEE 802.15.4 standard [Soc06].

In this section, I first show experimentally that CI really boosts concurrent transmission reliability in a significant way. Then, I investigate the one-hop delay of the state-of-the-art CI implementation Glossy [FZTS11] by measuring the Start Frame Delimiter (SFD) timing of the participating nodes. The goal is to check how good is the synchronization accuracy as well as the quantity and the reason for its variance. I derive a statistical model of one-hop delay, which can be applied to facilitate the protocol design and simulation based on CI.

5.1.3.1 *Experiment Setup for Studying Constructive Interference*

The experiment setup is shown in Fig. 5.1. The code is adapted from the publicly available Glossy implementation. An initiator node i and N responder nodes (all TelosB motes [PSC05]) are placed roughly 1 meter apart and communicate in 250ms lapses. A period starts when node i transmits a ping packet with the highest power

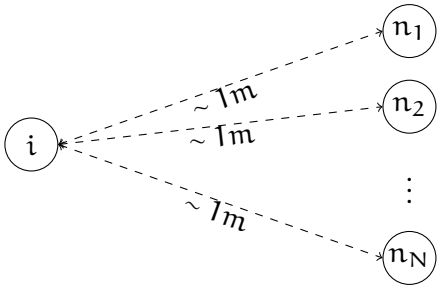


Figure 5.1: Experiment setup for studying Constructive Interference.

possible (given the short distance and line-of-sight, the responders almost certainly hear the packet). There are two types of periods: *concurrent tx period* and *individual tx period*. The first one accounts for three quarters of all cases, during which the responders (node n_1 to n_N) repeat the packet from node i concurrently. I call the repeated packet the pong packet. The individual tx period appears every second, during which each responder is asked to repeat in round-robin fashion while the others are required to keep silent. It takes N seconds until the individual *tx* period of a certain responder repeats. The reason for doing so is to measure the signal strength of each responder at the receiver continuously. In addition, the success or failure in reception of the concurrent transmissions and the noise floor are recorded at node i.

I use a logic analyzer of 24 MHz sampling rate to capture the SFD activity of the CC2420 radio chips of all nodes. According to the datasheet of CC2420 [CC213], in the receive mode, the SFD pin rises once the SFD byte has been completely received and falls only after the last byte of MAC Protocol Data Unit (MPDU) has been received. In transmit mode, the SFD pin rises when the SFD byte has been completely transmitted and falls when the complete MPDU has been transmitted. In general, I can take the SFD pin rise/fall as the start/end of packet transmission/reception. In each experiment, I sample the SFD pin of each node for 10^{10} times. The total time span is about 7 minutes [1], during which over 1600 rounds of ping-pong transmissions are performed.

5.1.3.2 *The Packet Reliability Boosts of CI*

Experiments have been carried out with 4 responders. I carefully adjust node position so that the responder signal strengths are similar at the initiator, falling in a small range from $-57dBm$ to $-60dBm$. In total, I have collected results for over 1200 concurrent transmissions of packet size 126 bytes. If there were no CI, the strongest transmission would be regarded as the signal and the others as the interference (the so-called *capture effect* will be elaborated later). In this case, I compute from Received Signal Strength Indicator (RSSI) readings that the SINR has a mean value of $-3.01dB$ and a negligible standard deviation of $4 \cdot 10^{-14} dB$. The SINR/PRR relation is depicted in Fig. 5.2 using the SINR/BER formula taken from the IEEE 802.15.4 standard (Eq. 5.8) [Soc06]. This relation predicts that the expected PRR is 0% while the measured one is 94%. This evidences unambiguously the occurrence of constructive interference and its capability of improving signal strength and thus PRR. Moreover, measurements confirm that

1 $\frac{10^{10}\,\text{samples}}{24 \cdot 10^{6}\,\text{samples/second}} = 416.667\,\text{seconds} = 6.94\,\text{minutes}$

the range of the transmission timing of all responders is never over 0.5µs in the short experiment period of about 7 minutes.

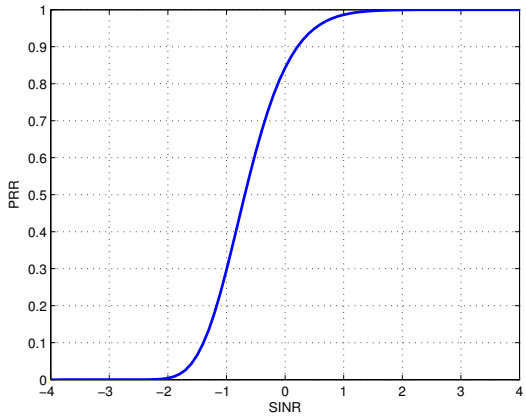


Figure 5.2: The SINR-PRR relation for IEEE 802.15.4 standard (when the packet length is 126 bytes).

5.1.3.3 *Modeling One-hop Delay of Glossy*

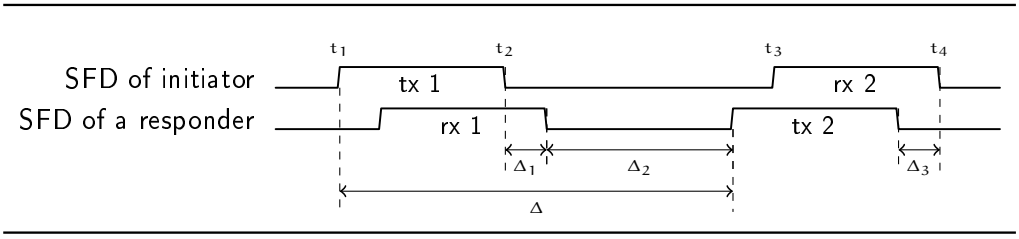


Figure 5.3: The SFD activities of a ping-pong round.

In this section, I model Glossy's one-hop delay as well as the resulting temporal displacement between two arbitrary nodes. The displacement determines whether the condition of constructive interference can be met or not.

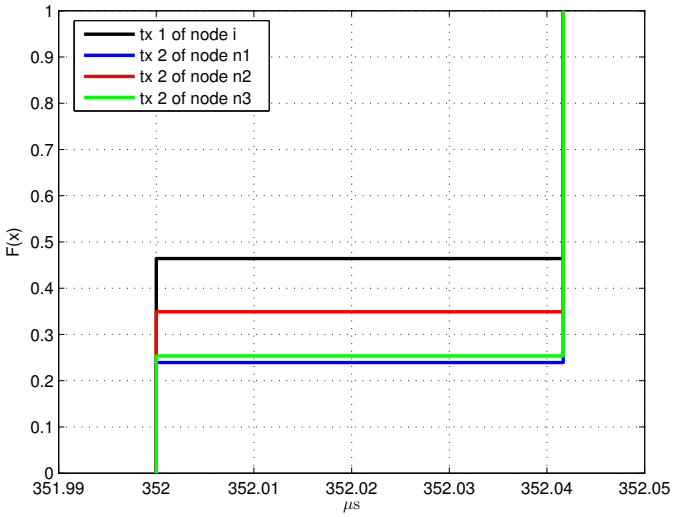
I investigate the problem with the experiment described in Sec. 5.1.3.1 with one initiator and three responders. The four motes are randomly chosen from a large set of motes. For each ping-pong round, I expect to observe the SFD activities as shown in Fig. 5.3. Four SFD level change times are recorded for each node (the initiator as well as the responders), namely t_1, t_2, t_3 and t_4. For the initiator, the four t_is represent the beginning and the end of the ping packet transmission and the beginning and the end of pong packet reception, respectively. On the contrary, for the responders they represent the beginning and the end of the ping packet reception and the beginning and the end of the pong packet transmission respectively [2]. Timings are measured with a logic analyzer of 24MHz sampling rate, which has a measurement precision of 41.7ns.

I model one-hop delay Δ by looking at its 3 components: the *tx* duration, the data latency and the software delay. The measurements are done for packet lengths of

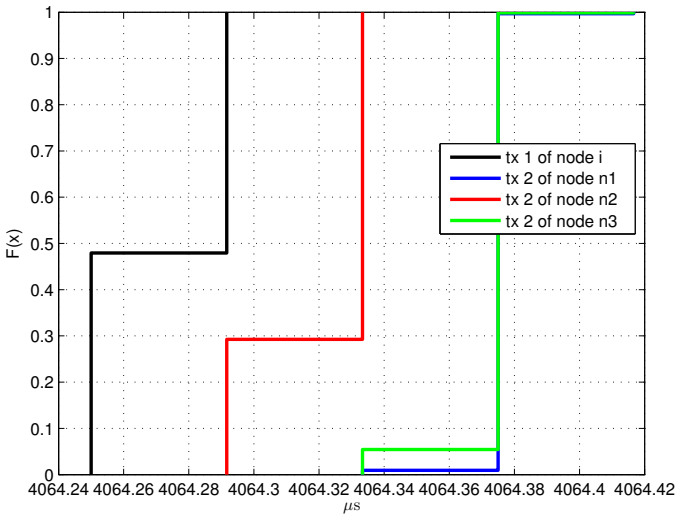
2 More accurately, the *rx/tx* begins 5 bytes before the SFD changes as I need to take the preamble and SFD byte into account.

10 and 126 bytes, respectively. Note that I ignore the propagation delay due to the short distance between the initiator and responders. But it can be easily added to the modeling.

5.1.3.4 *Transmission Duration* T



(a) CDF of transmission duration (10 bytes)



(b) CDF of transmission duration (126 bytes)

Figure 5.4: Distribution of transmission duration.

Time measurement for a $l+1$ byte transmission (l is the MPDU length, the time elapsed between SFD rising and falling is exactly $l+1$ bytes) is depicted in Fig. 5.4. Given a node and packet length, the transmission duration remains reasonably stable, as it is distributed in no more than 3 consecutive sample bins (the distance between 2 bins being 41.7ns). However across the nodes, there are large differences. Theoretically, the duration of a $l+1$ byte transmission should be $8(l+1)/250K = 32(l+1)\mu s$. But

actually, the oscillators of different radios have different degrees of frequency skew. From Fig. 5.4(a) and (b), I observe that the mean *tx* duration of node $i < n_2 < n_3 < n_1$ for both short and long packets, which shows the skew can be modeled as a constant for a node in a short period of time. So, I revise the model as $32(\frac{l+1}{1+k\cdot 10^{-6}})$, where k is the frequency skew measured in ppm. From the statistics of mean transmission time, I can calculate the constants k in Tab. 5.1.

Table 5.1: Frequency skew k of the radio oscillator of each node.

node	i	n_1	n_2	n_3
k (ppm)	-66.9	-92.2	-79.0	-91.7

Recall that the duration between SFD rising and falling measures the time for transmission of $l+1$ bytes. In addition, there are the *rx/tx* turnaround time (6 bytes according to [CC213]), the time for transmission of preamble (4 bytes) and SFD byte in each packet transmission before the SFD rising. Therefore, I model the transmission duration as follows:

$$T = 32\left(\frac{l+12}{1+k\cdot 10^{-6}}\right) \tag{5.1}$$

Thus, the difference of the *tx* duration D_T between two nodes with frequency skew k_1, k_2 is

$$\begin{aligned} D_T &= 32(l+12)\left(\frac{1}{1+k_1\cdot 10^{-6}} - \frac{1}{1+k_2\cdot 10^{-6}}\right) \\ &\approx 32(l+12)\left(\frac{(k_2-k_1)\cdot 10^{-6}}{1+(k_1+k_2)\cdot 10^{-6}}\right) \\ &\approx 32(l+12)(k_2-k_1)\cdot 10^{-6} \end{aligned} \tag{5.2}$$

The frequency skew has a deteriorating effect on CI, i.e., D_T goes up linearly with l, which means later bytes in the concurrent packets may have larger time displacement with respect to each other. It is possible that at the packet start, symbols add up constructively, but to the end, the CI condition is no longer met.

5.1.3.5 *Data Latency* L

I define data latency as the time difference between the end of packet transmission and the end of packet reception (Δ_1, Δ_3 in Fig. 5.3), i.e., the difference between the times that the SFD pins of the sender and receiver nodes fall. [3]

Fig. 5.5 depicts the CDF of the measured data latency Δ_1 and Δ_3. Δ_1 exhibits similar distribution for different responders within a small range of 0.21µs (5 sample periods). On the contrary, Δ_3 measured between nodes n_1, n_2, n_3 (as sender) and node i (as receiver) shows two types of distribution. Δ_3 measured with node n_1 has the same narrow distribution as Δ_1, while the other two have wider distribution. I argue the reason is that, in the experiment, irrespective of whether CI takes place, *capture effect*

3 Note, this definition is slightly different from the data latency definition in the CC2420 datasheet, where it is defined as the time from complete transmission of the SFD byte until complete reception of the SFD ignoring propagation delay. The typical value is 3µs.

always takes place, i.e., the receiver's radio synchronizes with the sender whose signal is the strongest heard, which happens to be always node n_1. This is specifically known as *power capture* [WWJ+05]. Even if the responders are placed next to each other and have the same *tx* power and the same distance to the initiator, the non-symmetric radiation pattern of the antenna and small-scale multi-path fading cause nonetheless different but stable reception powers. The capture effect is caused by the continuous SFD search implemented in CC2420 [CC213].

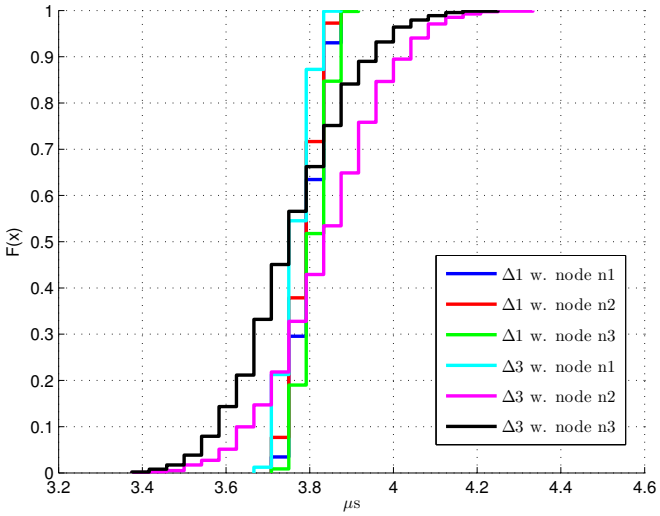


Figure 5.5: CDF of data latency Δ_1 and Δ_3 (when the packet length is 126 bytes). The plots when the packet length is 10 bytes are very similar.

Statistically, the data latency can be modeled as a random variable L, with mean value $\mu_L = 3.79\mu s$ and variance $\sigma_L^2 = 0.0019$.

5.1.3.6 *Software Delay S*

Software delay is defined as the time delay between the moment in which a node receives the entire packet and the moment in which it starts retransmitting the packet [4]. It is called compensated software delay (t_{sw}) in [FZTS11]). This is computed as $\Delta_2 = t_3 - t_2$ measured at the responders minus the time for transmitting 11 bytes (turnaround/calibration time, plus the transmission time for the preamble and the SFD byte). Although Glossy has tried very hard to make the software delay constant, it still has considerable randomness.

Fig. 5.6 displays that the software delay at the 3 responders follow a very similar distribution with mean value $\mu_S = 23.28\mu s$ and variance $\sigma_S^2 = 0.008$. The reason of the relative big standard deviation is due to the low clock frequency, which is equal to 4MHz, hence the time measurement and compensation precision is about 0.24 μs.

4 More accurately, the moment the radio starts calibrating Voltage Controlled Oscillator (VCO) before transmission.

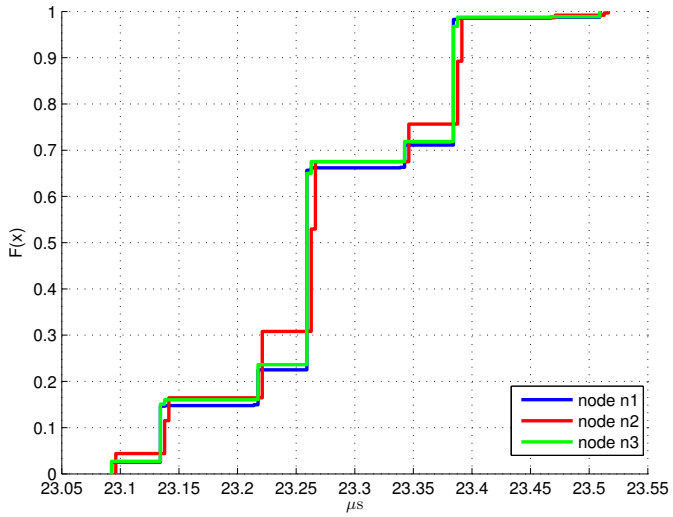


Figure 5.6: CDF of the software delay S (when the packet length is 126 bytes). The plots when the packet length is 10 bytes are very similar.

5.1.3.7 *One-hop Delay* Δ

Modeling the one-hop delay Δ of Glossy offers me clues on how well CI can be implemented. For a relay node, Δ is defined as the difference between the moment a packet starts being transmitted by an up-level node and the moment it starts being relayed. The up-level node is the one from which the highest signal strength is heard among all concurrent transmitters (e.g., in Fig. 5.3, Δ is equal to t_3 at a responder minus t_1 at the initiator).

The one-hop delay is the sum of transmission duration T, data latency L and software delay S.

$$\Delta = T + L + S = 32\left(\frac{l+12}{1+k\cdot 10^{-6}}\right) + X \tag{5.3}$$

where X is a random variable with mean value $\mu_X = \mu_L + \mu_S = 27.07\mu s$, and variance $\sigma_X^2 = \sigma_L^2 + \sigma_S^2 = 0.0099$. It is L and S lumped together, because they can be considered independent, after examination of the correlation coefficient. Hence, the difference of Δ between two nodes, $D_\Delta = D_T + D_X$. Furthermore, since X of two nodes are independent, I have

$$\begin{aligned} D_\Delta &= 32(l+12)\left(\frac{1}{1+k_1\cdot 10^{-6}} - \frac{1}{1+k_2\cdot 10^{-6}}\right) + Y \\ &\approx 32(l+12)(k_2-k_1)\cdot 10^{-6} + Y \end{aligned} \tag{5.4}$$

where Y is a random variable with mean value $\mu_Y = 0\mu s$ and variance $\sigma_Y^2 = 2\sigma_X^2 = 0.0198$, where k_1, k_2 are the frequency skews of the two nodes. The randomness of Δ mainly comes from the *tx* duration T and the software delay S, since the data latency variance is less than a quarter compared to S.

I assume the worst scenario possible. Suppose a topology such as the one depicted in Fig. 5.7. Nodes a and c transmit a packet at exactly the same time, b can only hear

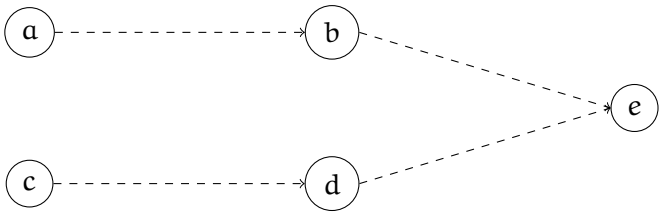


Figure 5.7: Topology to analyze worst-case timing for constructive interference in two hop scenario.

a, d can only hear c and e can hear both b and d. Node a and b have $k = 40ppm$, node c and d have $k = -40ppm$ (the CC2420 datasheet [CC213] specifies that the difference in frequency skew between two nodes is no larger than 80ppm), and $l = 127$ bytes. I want to study if CI happens at e, i.e., the two packets add up constructively.

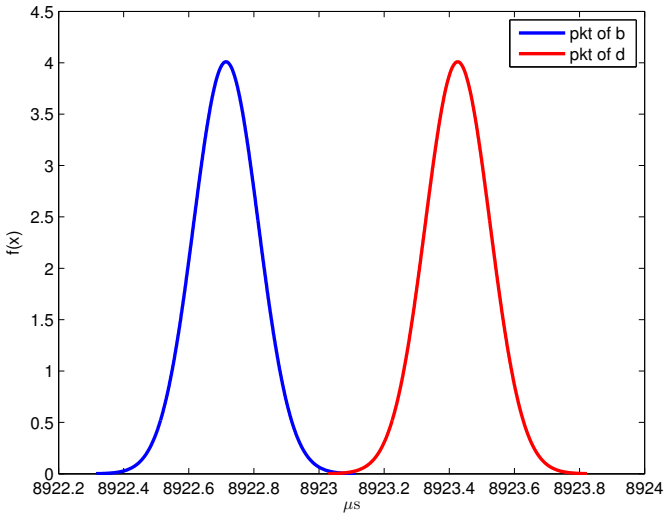


Figure 5.8: The distribution of transmission finish time of packets from b and d.

Suppose a and c start transmission at time 0, then the instant when b or d starts transmission can be modeled as a one-hop delay random variable (Δ), and the instant when b or d stops transmission as a random variable $\Delta + T$ (one-hop delay plus a transmission duration). Ignoring propagation delay, either packet arrives at e during these points in time. The distributions of the transmission ending time of packets from b and d at e are graphed in Fig. 5.8. We can guarantee constructive interference if the temporal displacement between packets from b and d are less than or equal to 0.5μs. The probability is only 6.6%.

Since the distance between the peaks of the two distributions in Fig. 5.8 is proportional to $(l + 12)(\frac{1}{1-k\cdot 10^{-6}} - \frac{1}{1+k\cdot 10^{-6}}) \approx 2(l + 12) \cdot k$, either a small k or a small l will improve the probability of CI (the two distributions move to the figure center).

In summary, it is hard to guarantee CI, when the packet is long and the oscillators have extreme frequency skews. Therefore, to reduce the variance due to *tx* duration T, I have to use small packets, because I normally have no control on the frequency skew of the radio chip, which is determined by manufacture and temperature. To reduce the variance due to software compensation, I may use Microcontroller Units (MCUs) that

have clock of higher frequency and better stability. This allows for higher compensation precision and accuracy. In addition, as proposed in [WHC+13], I can use the new generation chip integrating MCU and radio because the synchronized clock will eliminate the communication time uncertainty between the two modules. Fortunately, the capture effect helps to improve packet reliability and it leads to successful reception when the strongest packet is higher than the sum of the others over a certain degree (e.g., $2dB$ for the IEEE 802.15.4 radio, as shown in Fig. 5.2). Moreover, one node having higher power than the others in a number of concurrent transmitters has the advantage that the receivers will likely synchronize to the same node, thus avoiding the accumulation of one-hop delay.

5.1.4 *Capture Effect*

Capture effect is the ability of the radio to receive a strong signal despite interference from other concurrent transmissions. When the stringent time synchronization condition of less than $0.5\mu s$ cannot be met, it is still very probable that capture effect happens since it has much looser time synchronization requirement. Namely, the time synchronization of capture effect needs to be only less than the preamble duration T_p ($128\mu s$ for the IEEE 802.15.4 radio) as shown below.

5.1.4.1 *Experiment Setup for Studying Capture Effect*

The differences in the experiment setup to that of Sec. 5.1.3.1 are: a) after the responders receive the ping packet, they delay it for a random time uniformly distributed between 0 to $360\mu s$ before resending; b) in this experiment, ideally I want to measure the arrival times of all concurrent pong packets. However, this is almost impossible since all these packets are *mixed up*. Given the fact that the distances between the initiator and the responders are small (about 1 meter) and similar to one another, I can equivalently measure the transmission start times.

5.1.4.2 *Two Methods for Measuring Transmission Start Time*

I propose two possible methods for measuring transmission start time: one is hardware-based and the other is software-based. The first one uses a logic analyzer to measure SFD level changes. However, using this method might not be feasible when the responders are located far from each other or the number of responders is high [5]. In such cases, a software-based method is preferred.

I have developed a software method to capture *tx* start and *rx* end times taking advantage of the fact that the SFD pin of the CC2420 chip is connected to timer B of the MSP430F1611 MCU [Tex11] of the TelosB motes. Then, at each responder, I record the time difference between the end of the ping packet reception and the start of the pong packet transmission associated with a packet sequence number. Conceptually, it can be regarded as the *tx* start time because the responders should get the packets at almost the same time.

To get a high measurement precision, I source timer B with the 4MHz Digitally Controlled Oscillator (DCO), the main clock of the MCU, which gives a measurement

5 A logic analyzer has limited number of channels.

precision of approximately 0.24µs. But the DCO is very sensitive to temperature and voltage change. To mitigate its frequency skew, I calibrate it periodically with the stable 32KHz crystal on the board after each ping-pong round.

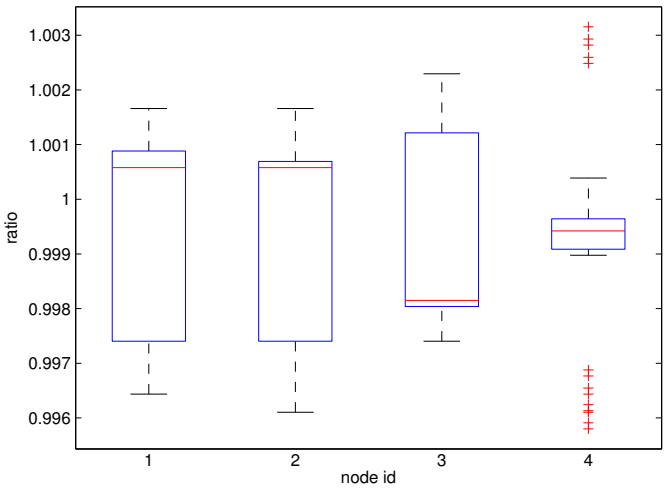
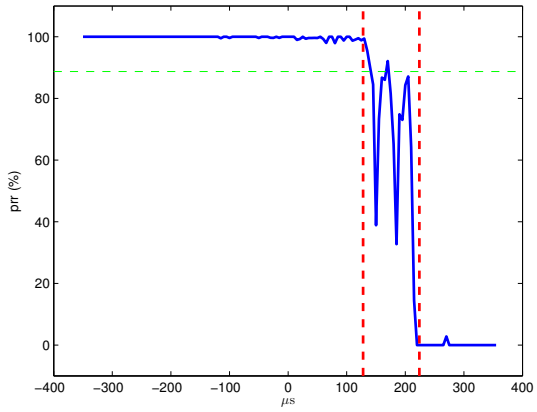


Figure 5.9: Time measurement comparison between software and hardware methods (ratio = software reading / hardware reading).

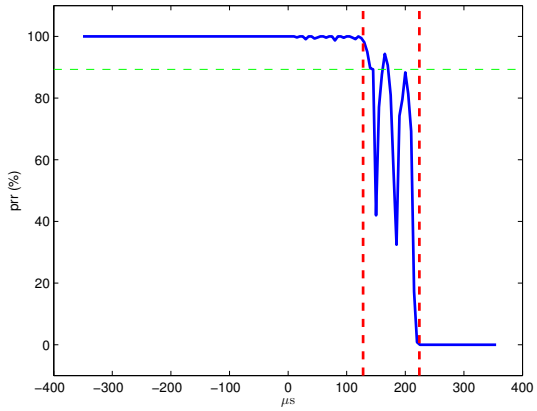
In order to evaluate the quality of the software method, I perform the ping-pong experiment for over 1200 times and measure the time difference between the end of the ping packet reception and the start of the pong packet transmission at the responders with both methods. The results for 4 nodes are shown in Fig. 5.9. I observe that 50% of the measurements have an error not bigger than 0.3% and in the worst case, the measurement has an error of the order of 0.4%. Because the time difference between packet reception and retransmission (including the intentional delay up to 360µs) is less than 740µs, the error should be within ±2.96µs. This accuracy is not suitable for measuring constructive interference, but it is enough for measuring capture effect.

5.1.4.3 *The Relation between Transmission Timing and Capture Effect*

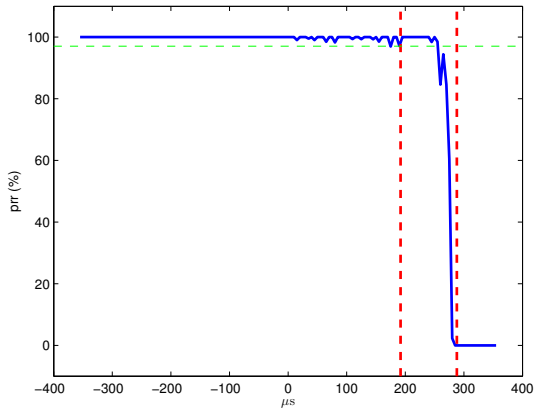
In this section, I show how the *tx* timing influences capture effect and thus the PRR. Fig. 5.10 shows the results corresponding to 3 runs of the 2-responder experiment. The packet length is short (10 bytes) or long (126 bytes), the preamble length L_p is 4 or 6 bytes. In each run, I perform 30,000 rounds of ping-pong transmission. Ideally, I want to measure the signal strength of each responder at the time of concurrent transmission, but this is impossible since signals are added up. However, in the quiet 802.15.4 channel #26 band and the quasi-static office environment, the signal strength of a responder can be regarded as slowly varying. Thus, I substitute the signal strength of a certain responder at the time of concurrent transmission with the mean value of signal strengths measured during the individual *tx* periods within ±30 seconds. The noise floor is the mean noise measured within ±1 second. The x-axis of Fig. 5.10 represents the difference in *tx* time of node n_1 and n_2, i.e., $\Delta_t = t_{n_1} - t_{n_2}$. The y-axis represents the PRR, calculated for every 5µs bins.



(a) $\overline{R_1} = -41.0, \overline{R_2} = -46.2, L_p = 4, l = 10$



(b) $\overline{R_1} = -41.0, \overline{R_2} = -48.2, L_p = 4, l = 126$



(c) $\overline{R_1} = -41.7, \overline{R_2} = -47.3, L_p = 6, l = 126$

Figure 5.10: The relation of *tx* time difference $\Delta_t = t_{n_1} - t_{n_2}$ and PRR, $\overline{R_i}$: mean RSSI (*dBm*) measured at node n_i, L_p: preamble length in bytes, l: packet length in bytes.

Given the preamble length $L_p = 4$ bytes, different packet lengths exhibit the same behavior (Fig. 5.10(a) and (b)). When $\Delta_t \leqslant 128\mu s$ (4 bytes), the PRR is almost perfect and when $\Delta_t > 128\mu s$, the PRR decreases to 0 in about $96\mu s$ (3 bytes). Besides, preamble length is set to 6 bytes and the experiment is performed again. In this case, the perfect PRR region expands over $\Delta_t = 192$ (6 bytes), but it still falls to 0 in 3 more bytes.

My explanation for these results is the power capture effect and n_1 has higher signal strength than n_2 ($R_1 > R_2$). When $\Delta_t \leqslant 0$, the packet from n_1 is received due to the *stronger-first capture effect* [WWJ+05], i.e., the radio at i is synchronized with the earlier and stronger packet from n_1. However, for the CC2420 radio, I observe that the capture effect extends at least to $\Delta_t = T_p$, i.e., the strong signal comes no later than the preamble duration after the weak signal. This is because the CC2420 radio continuously synchronizes with the zero-symbols in the preamble and searches for the SFD byte [CC213]. It resynchronizes to a stronger signal before it detects the SFD. A stronger signal coming after the SFD detection may not lead to resynchronization. If so, it will lower the SINR and increases the probability of packet loss.

In summary, if the *tx* time of the strong signal minus that of the weak signal is less than or equal to the preamble duration, i.e., $\Delta_t \leqslant T_p$, then the capture effect is sure to occur. The capture effect also occur to some degree if $T_p < \Delta_t < T_p + 96\mu s$. When the strong signal comes later than the weak signal for over $L_p + 3$ bytes, i.e., $\Delta_t \geqslant T_p + 96\mu s$, the capture effect does not occur.

5.1.5 *A Packet Reception Model for Concurrent Transmission*

To build a packet reception model, I need to understand how the process of concurrent transmission works. Whether a packet is received or not is the consequence of the combined result of the following three effects:

1) Capture effect. When the first packet arrives, the receiver's radio synchronizes with it. Within the preamble duration (conservative estimation), if a packet with a higher signal strength than the *sync packet* (the packet to which the receiver's radio has synchronized) arrives, it will capture the radio. This implies that the radio re-synchronizes to the new packet and the preamble duration at the receiver restarts. In the case of n concurrent packets, the maximum preamble length at the receiver can be as large as $n \cdot T_p$ bytes, when the packets arrive in the separation of preamble duration T_p and the later packets have higher signal strength than the previous. Being conservative, I assume any packet coming after the preamble duration will not capture the radio even if it had a higher strength than that of the sync packet.

2) Constructive interference. If a packet has the same content as the sync packet and the time displacement between these two at the receiver is less than or equal to the half of chip duration T_c, then they add up constructively, thus increasing the signal strength.

3) SINR/SER model. The symbol error is probabilistically determined by the SINR. The packets that form CI with the sync packet, add to the signal by a factor depending on the temporal displacement while those that do not, add to the interference, which reduces symbol error rate. In addition, the background noise should also be considered.

Algorithm 12: The packet reception model of concurrent transmission.

Data: $\mathbb{O}$: node set; $t(n_i)$: the *tx* start time of node n_i; $s(n_i)$: the signal strength of node n_i; f: the noise floor; $\mathcal{M}$: the SINR/SER model.
Result: A boolean value indicating whether a packet is received or not.

sort $\mathbb{O}$ in ascending order by $t(n_i)$. Suppose $\mathbb{O} = \{n_1, ..., n_N\}$ in order;
// search for *sync node*
sync node $c = n_1$, preamble time $p = t(n_1)$;
for $i = 2$ **to** N **do**
    if $t(n_i) - p \leqslant T_p$ **then**
        if $s(n_i) > s(c)$ **then**
            $c = n_i, p = t(n_i)$;
    else
        break; // no more capture
assign nodes from n_1 till c to *signal set* $\mathbb{S}$ or *interference set* $\mathbb{I}$ depending on $t(c) - t(n_i) \leqslant 0.5 \cdot T_c$;
calculate SINR with Eq. 5.6 and Eq. 5.7, $SER = \mathcal{M}(SINR)$, $PRR = 1$;
for n_i *in* $\mathbb{O}$ *after* c **do**
    calculate $t(n_i) - t(n_{i-1})$ in number of symbols m, $PRR = PRR \cdot (1 - SER)^m$;
    assign node n_i to $\mathbb{S}$ or $\mathbb{I}$ depending on whether $t(n_i) - t(c) \leqslant 0.5 \cdot T_c$;
    update SINR, SER;
set *tx* end time for each node $e(n_i) = t(n_i)$+ *packet duration*;
for $n_i = n_1$ **to** c **do**
    update PRR till time $e(n_i)$;
    remove n_i from $\mathbb{S}$ or $\mathbb{I}$;
    update SINR, SER;
return *a random number* $r \in [0, 1) <$ PRR;

Putting these three effects all together, I propose a model/algorithm [6] capable of predicting the result of packet reception of concurrent transmission, which is listed in Alg. 12.

The algorithm is efficient in terms of computation time since it is of the order of $O(N)$, N being the number of concurrent transmitters. I use the RSSI measurements to calculate SINR. For a node n_i, its signal strength is

$$s(n_i) = 10^{RSSI_i/10} - 10^{f/10} \tag{5.5}$$

where $RSSI_i$ is the RSSI measurement when only n_i is transmitting. I use the mean RSSI over ± 30 seconds for higher accuracy. f is the noise floor measurement. I use the mean noise floor over ± 1 second. To compute the signal strength of the *signal set* $\mathbb{S}$, given the sync node c (the set contains all nodes whose packets add up constructively with c, including c), I use the equation

$$s(\mathbb{S}, c) = \left[\sum_{n \in \mathbb{S}} \sqrt{s(n)} \cdot \cos\left(\frac{t(n) - t(c)}{T_c} \cdot \pi \right) \right]^2 \tag{5.6}$$

6 The model assumes that the concurrent packets are overlapping, i.e., the last packet starts before the first packet ends, in order not to over-complicate the description. It is trivial to extend the model for the other case.

where $t(n)$ is the arrival time of the packet from node n. The reasoning draws mainly on the waveform analysis outlined in [WHM+12]. Then SINR is

$$\text{SINR} = \frac{s(S, c)}{\sum_{j \in \mathbb{I}} s(n_j) + 10^{f/10}} \tag{5.7}$$

where $\mathbb{I}$ is the *interference set*, the complement set of S.

In the validation of the model, I use the SINR/SER relation deduced from the SINR/BER relation in IEEE 802.15.4 standard [Soc06].

$$\text{BER} = \frac{8}{15} \cdot \frac{1}{16} \cdot \sum_{k=2}^{16} (-1)^k \binom{16}{k} e^{20 \cdot \text{SINR} \cdot (\frac{1}{k} - 1)} \tag{5.8}$$

$$\text{SER} = 1 - (1 - \text{BER})^4 \tag{5.9}$$

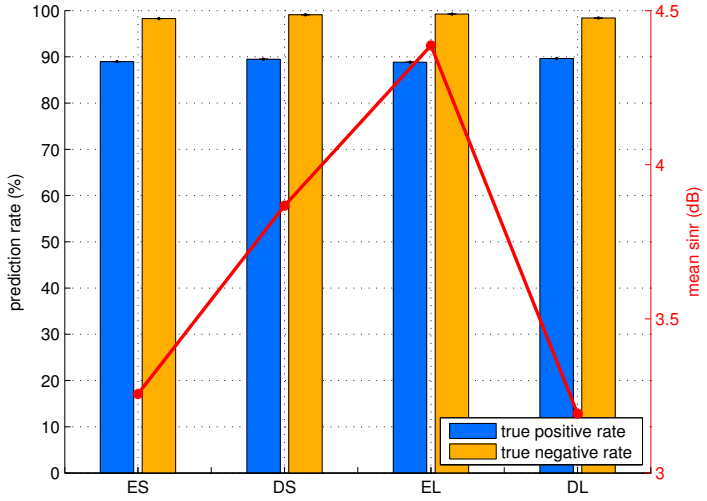
To test the prediction quality of my packet reception model, I assess it with eight experiments of over 30,000 transmission rounds each. These experiments are performed for 2 and 6 concurrent transmissions with either same (level 31) or different *tx* powers (for 2 responders, the powers are 31 and 11 respectively; for 6 responders, the powers are 31, 27, 23, 19, 15, 11 respectively) [7] and with long (126 bytes) or short (10 bytes) packet length. The results are depicted in Fig. 5.11.

I define as positive when a successful reception has been predicted as such. Thus, true positive rate is the percentage of actually received packets that have been predicted as received while true negative rate is the percentage of actually lost packets that have been predicted as lost.

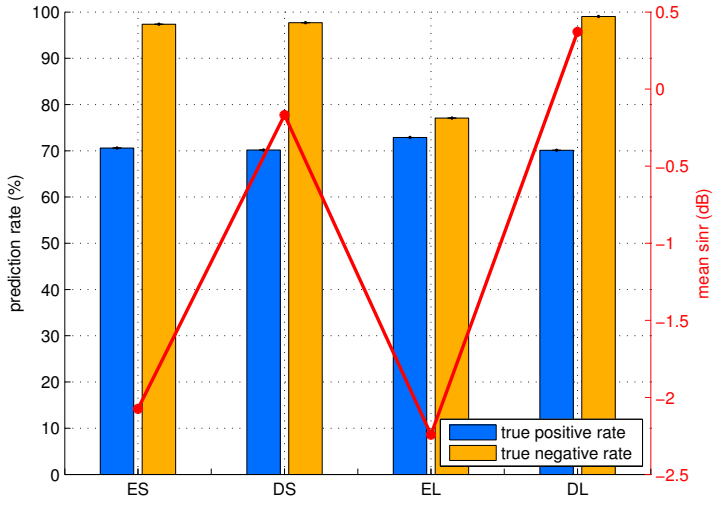
$$\begin{aligned} \text{true positive rate} &= \Pr(\text{packet predicted as received} \mid \text{packet actually received}) \\ \text{true negative rate} &= \Pr(\text{packet predicted as lost} \mid \text{packet actually lost}) \end{aligned} \tag{5.10}$$

My model exhibits a high prediction quality, having a true positive and true negative rate over 70%. Furthermore, it has higher accuracy in predicting lost packets than received packets and, in many cases, it has almost 100% true negative rate. This is due to the conservative way capture effect has been modeled, i.e., capture effect only happens within preamble duration of the sync packet. The prediction quality for two concurrent transmissions (true positive/negative rate is greater than or equal to 90%) is better than that for six transmissions. It is due to the inaccuracy of the SINR/SER model, especially in the transitional region (SER change from 0 to 1). SINR are more often encountered in this transitional region in the six-transmitter setting than in the two-transmitter one. This can be confirmed by checking the mean SINR value of concurrent transmissions depicted in Fig. 5.11. Here, mean SINR is greater than $3dB$ for the two-transmitter setting and it is under $0.5dB$ for the six-transmitter one. SINR is computed by considering those nodes in S as the signal node and the others as interference nodes. Also, the imperfect prediction quality might be caused by the inaccuracy in RSSI reading [CT11].

7 The power levels 31, 27, 23, 19, 15, 11 correspond to transmission power $0, -1, -3, -5, -7, -10$ dBm respectively [CC213].



(a) 2 concurrent transmissions



(b) 6 concurrent transmissions

Figure 5.11: Prediction quality of 2 and 6 concurrent transmitters. E/D stand for equal/different *tx* power. S/L stand for short/long packet.

5.1.6 *Conclusion*

With my work, I provide a comprehensive understanding of the concurrent transmission in WSNs. I give theoretical explanations on the nature of constructive interference and capture effect, and validate these experimentally.

I model the one-hop delay of the state-of-the-art CI implementation Glossy and show the problem in achieving CI using commodity sensor nodes. My measurements show that the temporal displacement of one-hop delay between two arbitrary nodes is a random variable with standard deviation $0.14\mu s$ and mean value which can be as high as $0.36\mu s$ in the worst case for a packet length of 127 bytes (cf. IEEE 802.15.4) and a relative frequency skew of $80ppm$ between the radio's oscillators (cf. CC2420). I can conclude that CI is not easy to obtain with the currently available commodity sensor nodes, since it requires the temporal displacement between transmission to be lower than $0.5\mu s$. However, I expect that a hardware based implementation of retransmit will fully eliminate the software delay. Thus, the one-hop CI can be guaranteed. The extend to which the temporal displacement accumulates with the number of hops in real-world applications is waiting to be investigated.

In contrast, successful reception due to the capture effect is far easier to achieve because it only requires that the strong signal arrives no later than the preamble duration ($128\mu s$ for the IEEE 802.15.4 radio) after the weak signal and a sufficiently high SINR.

The accurate reception prediction model proposed provides a complete toolset for predicting concurrent transmissions for a single hop and a multi-hop network. The models and insights gained are valuable for simulation and protocol design of concurrent transmission in WSN.

5.2 SPARKLE: A NETWORK DESIGN EXPLOITING CONCURRENT TRANSMISSIONS FOR ENERGY EFFICIENT, RELIABLE, ULTRA-LOW LATENCY COMMUNICATIONS IN INDUSTRIAL CONTROL

5.2.1 *Introduction*

Wireless sensor networks offer great potential, and yet also great challenges for industrial automation and control applications. Despite the fact that some pioneering works in academia and a few industrial standards have appeared, to the best of my knowledge, no large scale application of WSN in industrial automation has yet taken place.

Recently, the Glossy protocol [FZTS11] showed the possibility of obtaining deterministic low latency, high reliability and high synchronization precision simultaneously by applying the technology of CI. These features match the requirements of control networks so good that Glossy can offer a sound basis for it. Based on Glossy, I design and implement *Sparkle*, a periodic multi-loop control network where each control loop is mapped into one or more communication flows (same as I did in Sec. 4.2.). The novelty of Sparkle is that I perform "control" on each flow (say flow $a \rightarrow b$, a and b being two end nodes), with the goal that the QoS (quality of service) metrics of the flow satisfy given requirements or are optimized. For that purpose, I need the

cooperation of the opposite flow (flow $b \rightarrow a$) to perform feedback control based on the measurements at the destination.

Specifically, I show that by effectively controlling the network topology and transmission power (tx power) of a flow, the QoS metrics of reliability, energy consumption and latency can be further improved simultaneously, compared to Glossy. I propose a novel technique for topology control, *WSNShape*, which uses the *capture effect* [WWJ+05] to find a number of reliable paths between the source and the destination of a flow and then activate nodes on one or more of these paths. As shown by evaluation, it greatly reduces energy consumption and very probably also improves end-to-end reliability and latency. Additionally, I experimentally show that the transmission power also affects the QoS metrics significantly and the Glossy protocol without WSNShape may not be reliable enough for control networks.

Based on these findings, I design and implement the "controller" of Sparkle, *PRRTrack*, which adaptively switches between operation modes of different transmission powers and WSNShape levels. Experiments on two real-world testbeds show that the requirement on reliability is satisfied, the latency is reduced, and the energy consumption is greatly improved over today's state-of-the-art techniques.

5.2.2 *Related Work*

Constructive interference, i.e., the concurrent transmission of the *same* packets by multiple transmitters can achieve almost optimal latency, high reliability and time synchronization of μs accuracy [FZTS11]. Since these features match the requirements of automation and control systems perfectly, I choose Glossy as the starting point to design my flow-based communication system Sparkle. When concurrent transmissions of *different* packets take place, one packet may overpower the others and be successfully received. This is called capture effect. The capture effect in WSN was first discussed by [WWJ+05]. A recent work, Chaos [LFZ13a] shows the universality of the capture effect in WSN and employs it to implement an efficient all-to-all aggregative communication. WSNShape, the topology control technique of Sparkle, also makes use of the capture effect, and is therefore instantly reactive to node or link failure.

To optimize the energy efficiency of the end-to-end communication primitive implemented with the Glossy flooding technique, [CCT+13] uses the hop count from the source node as a metric for forwarder selection (topology control). It is shown to reduce the energy consumption by 30%. In contrast, I use the capture effect for forwarder selection, which leads to energy savings of about 80% to 90% together with better reliability and lower latency in networks of similar size. In addition, I assume nothing about the symmetry of the network (e.g. the hop count from node a to b is the same as from b to a), and the two directional flows between a pair of nodes are independently controlled.

In all, as far as I am aware of, Sparkle is the first wireless control network based on the technology of concurrent transmission [FZTS11][YH13], resulting in a system that has excellent performance in communication reliability, latency and energy consumption, as well as unprecedented robustness to node or link failure.

5.2.3 *The Architecture of Sparkle*

Sparkle employs a protocol similar to TDMA, which is normally used by wireless control networks, as it allows for deterministic scheduling and relatively deterministic QoS performance. The architecture makes independent QoS control of each end-to-end flow possible.

5.2.3.1 *Mapping the Communication of Control Systems to Flows*

Sparkle supports arbitrary communication requirements of periodic multi-loop control systems. For the simplest case of a Single Input and Single Output (SISO) control loop, it requires that periodically a packet with sensor data is transmitted from a sensor to a controller, and then a packet with actuation data is transmitted from the controller to an actuator. If I implement the controller on either the sensor, or the actuator, then I only need to maintain one QoS conformable flow, i.e., from the sensor to the actuator. On the other hand, if I implement the controller on a separate node, then two flows need to be maintained, i.e., from the sensor to the controller and from the controller to the actuator. For the more complex case of Multiple Inputs and Multiple Outputs (MIMO) control loop, I need to maintain multiple flows from every sensor to the controller and from the controller to every actuator. Besides, Sparkle also supports the communication of data collection and dissemination commonly required by monitoring applications. As mentioned before, to control a flow, I need the cooperation of the opposite flow for delivering control commands.

5.2.3.2 *Frame Structure*

A Sparkle frame is composed of a *sync slot*, a number of *data slots* and zero or one *control slot* (Fig. 5.12). In each slot, a flooding is performed with a source node (aka. initiator), a transmission power and a set of participating nodes.

sync slot	data slot #1	data slot #2	.....	control slot/idle

Figure 5.12: A Sparkle frame.

The purpose of sync slots is to obtain network-wide time synchronization, in which an authority node (normally located in the network center for a short average distance to all other nodes) floods a short sync packet over the network with the Glossy protocol. Since a sync packet is very short (10 bytes in our implementation), it has a very high chance of being correctly received by all nodes, if I use a proper transmission power. The network-wide time synchronization is a prerequisite for the data communication in Sparkle. The next data slots are used for the communication of arbitrary flows. Different flows may have different period length, dependent on the requirement of the control system. The control slots are used for the QoS control of the flows. Whether a set of flows (with arbitrary period length) are schedulable in Sparkle can be easily determined by the Earliest Deadline First (EDF) scheduling algorithm, which gives the optimal solution for the Sparkle scheduling as it can be mapped to a uniprocessor scheduling problem [Liu00]. A necessary condition of the schedulability is that the total utilization of all slots is no more than 1.

5.2.3.3 *Controlling the QoS Metrics of a Flow*

Sparkle is capable of performing different QoS control for different flows. Normally, the QoS controller of a flow is located at the destination node. It keeps track of the QoS metrics of the flow and sends out control commands to the source node or the whole network in the control slots of the opposite flow when necessary (e.g., setting transmission power or activating/deactivating nodes for a flow). This design decision has the advantage of easy implementation and independent performance control for each flow, even for a pair of opposite flows. The detailed control scheme of Sparkle will be expounded later.

5.2.4 *How Network Parameters Affect Performance*

In this part, I investigate experimentally in real-world testbeds how the transmission power and network topology affect the QoS metrics of reliability, latency and energy consumption. The so-called WSNShape technique is a novel topology control method that uses the capture effect to find reliable paths from the source to the destination of a flow. It is very effective in finding reliable paths, compatible with the Glossy protocol as it requires no unicast transmission, and is much more lightweight and faster than ordinary routing protocols. Furthermore, it is resilient to node failure, which is not provided by the normal routing protocols.

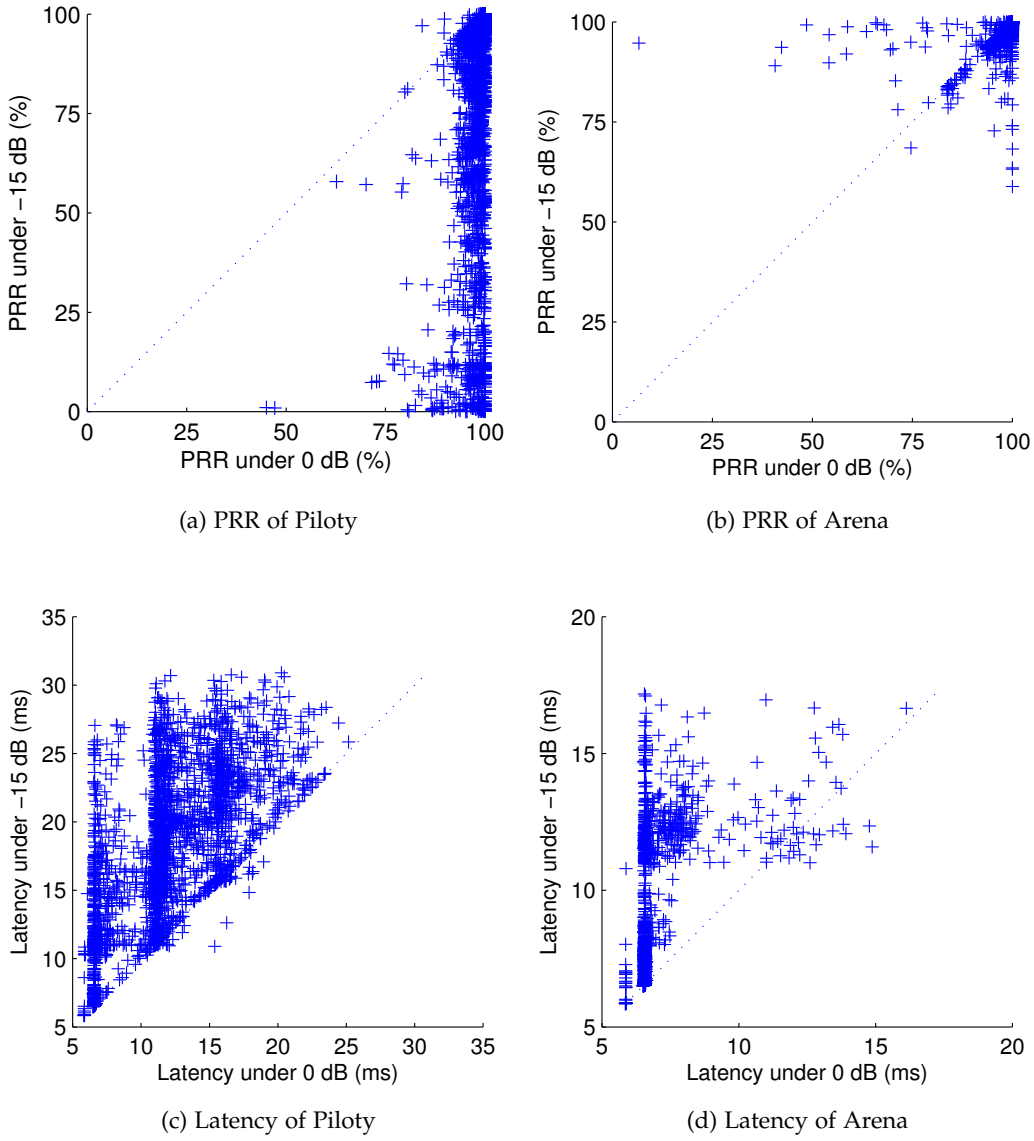
5.2.4.1 *Different Transmission Powers*

The evaluation results in the Glossy paper [FZTS11] indicated that a higher transmission power gives lower latency and higher reliability. However, my evaluation on the two TUDμNet testbeds [GBKVL12] shows that a higher transmission power may lead to lower reliability when the network connectivity is very high.

TUDμNet includes two testbeds called Piloty and Arena respectively. The former has 63 TelosB nodes [PSC05] (n_1 to n_{63}, 55 are active), located in various offices on two floors of the CS building at TU Darmstadt, spanning a volume of $30 \times 20 \times 8\,\mathrm{m}^3$. The latter has 60 TelosB nodes (n_{1001} to n_{1060}, 42 are active), forming a 5×12 grid, located in a large room with line-of-sight between any two nodes, spanning a volume of $31 \times 7 \times 3\,\mathrm{m}^3$. To compare different powers fairly, I let a source node perform Glossy flooding to all other nodes in the network, by setting the transmission power of all nodes to $0\mathrm{dBm}$ and $-15\mathrm{dBm}$ alternately per second. In this way, I exclude the effect of the relatively slow channel variation of static WSN. Each active node acts once as the source.

My results in Fig. 5.13 show that in the Piloty testbed, in almost all cases, the PRR is better when the higher transmission power is chosen. However, in the Arena testbed, quite often the higher transmission power gives lower reliability. The conclusion of the Glossy paper that the latency is lower under a higher transmission power is generally supported by my experiments [FZTS11]. Intuitively, it is due to the smaller hop count.

I argue that the reason that very often a lower transmission power improves PRR in the Arena testbed is a higher network connectivity than that in the Piloty testbed. For instance, in one experiment run in the Arena testbed, I found that at $0\mathrm{dBm}$, in average 33.04 nodes are one hop away from the source, while at $-15\mathrm{dBm}$, only 19.25 nodes are one hop away. Furthermore, one node has a PRR of 79% and 99% at $0\mathrm{dBm}$



(a) PRR of Piloty

(b) PRR of Arena

(c) Latency of Piloty

(d) Latency of Arena

Figure 5.13: The relation between PRR and end-to-end latency at different transmission powers. A point in the scatter plot is the metrics at 0dBm and −15dBm, for a given source and destination pair.

and −15dBm, respectively. At 0dBm, the node is for most of the time (62%) two hops away, hence it suffers from the low SNR caused by the large number of concurrently transmitting nodes at hop one. In contrast, that 0dBm has generally better PRR than −15dBm in the Piloty testbed is due to a lower connectivity when compared to that of the Arena testbed. The node density of the former per m^3 is only one ninth of the latter, and the separation of walls and floors further reduces the connectivity. This brings to light that anticipating a proper transmission power with respect to packet reliability is very hard. To do so, predicting the channel condition and taking the reception model of concurrent transmissions into account would be required (cf. Sec. 5.1). One practical way of overcoming this problem is to empirically determine

the transmission power. Finally, the phenomenon that a high transmission power may cause low reliability also evidences that topology control is necessary.

5.2.4.2 *Network Shaping with WSNShape*

Control networks normally feature one-to-one communication, which is a special case of the one-to-all communication intrinsically supported by Glossy. If I could find a stripe of nodes between the source and destination for a flow, and only perform Glossy flooding among these nodes, energy consumption would be significantly reduced since lots of nodes are deactivated, and hopefully there would be still enough nodes in the stripe to take the advantage of the high reliability of constructive interference. However, I face two obstacles in *network shaping*, i.e., how to find the stripe: 1) Glossy is a routing free protocol and derives its advantages in reliability and latency from this feature. Therefore, network shaping with traditional routing protocols is incompatible with Glossy. 2) Since the channel condition is time-variant, I should continuously perform network shaping, which requires the process to be very lightweight and fast. My novel WSNShape technique overcomes the two obstacles by effectively utilizing the capture effect.

Path Identification The most important step of WSNShape is path identification, i.e., to find the reliable paths between the source and destination of a flow. I use the control slots of the flow for this purpose (Fig. 5.12). The process is as follows:

1. Activate all nodes in the network.
2. The source sets the bit corresponding to itself in the *path-ident* packet and broadcasts it. The path-ident packet contains basically a bit set of N bits where N is the number of nodes in the network [8].
3. Any node relays the packet exactly once in the way as Glossy. One difference is that instead of rebroadcasting the packet unmodified, the node sets the bit corresponding to itself in the path-ident packet before rebroadcasting.
4. If a packet is correctly received at destination, the packet can be used to reconstruct a path from the source to the destination.

Path Identification is Effective and Lightweight

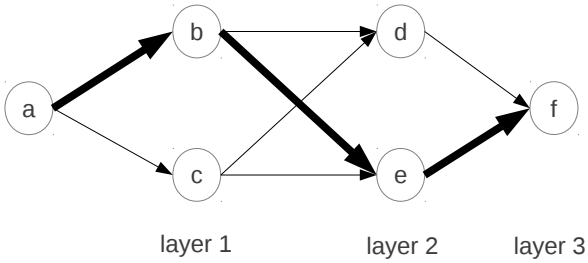
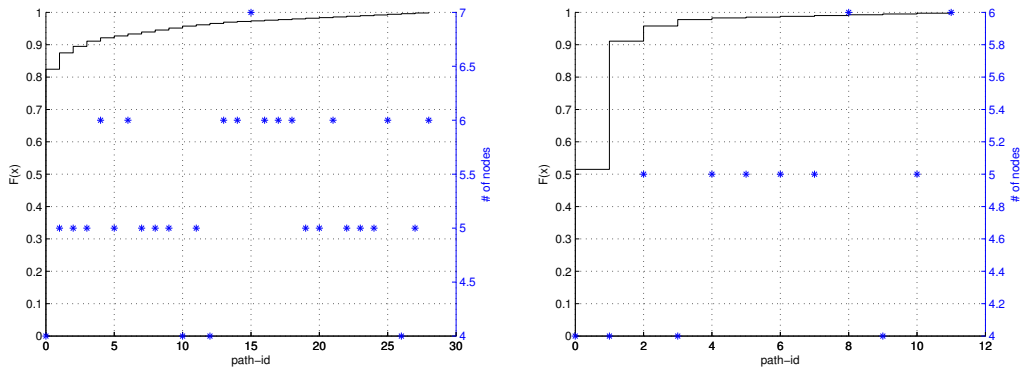


Figure 5.14: An example of path identification. $a \rightarrow b \rightarrow e \rightarrow f$ is an identified path.

8 A control network normally has less than 100 nodes which takes only a dozen of bytes. If the network size is much bigger than the path length, I should enumerate the node ID of each hop rather than using a bit set.

The capture effect implies that if a number of nodes transmit different packets concurrently and a packet is correctly decoded by the receiver, the packet should come from the node whose signal is the strongest at the receiver. Given that all nodes transmit with the same power, theoretically, I can infer that if a path is identified with the above process, then every link of the path, say $x \to y$, has the smallest signal loss among all links going into the node y, and the path has the shortest number of hops from the source to the destination.

Conceptually I build a network of K layers, where layer 0 only has the source node, and layer i contains all nodes that receive the packet after i relays. Each node in layer $i-1$ has a directional link to each node in layer i. Fig. 5.14 shows a path identified from source a to destination f in an example network. Then, the signal loss of $b \to e$ is smaller than $c \to e$, because node b and c are both one hop away from a and transmit simultaneously. Similarly, the signal loss of $e \to f$ is smaller than $d \to f$. The shortest number of hops can be obtained taking into account the network is layered and directional. Although I cannot say that the identified path has the globally smallest cumulative path loss, as each link is locally optimal, practically, the path should be reliable and short.



(a) flow $n_{33} \to n_{60}$, $P = 99.6\%$ (b) flow $n_{60} \to n_{33}$, $P = 99.5\%$

Figure 5.15: The CDF of the identified paths and the number of nodes in each of them, P is the path identification rate. Tx power = 0dB.

To evaluate the quality of path identification, I test it on a number of flows with source and destination far apart on two floors in the Piloty testbed. A control slot is applied for each flow every 6s. The results for flows $n_{33} \leftrightarrow n_{60}$ are shown in Fig. 5.15, which are similar to that of the other flows. The path identification is very effective, both flows have a path identification rate P (the percentage of times that a path is successfully found between two nodes for all path identification tries) of over 99%. This confirms that the capture effect is universal in WSN [LFZ13a]. This has the advantage that the path identification is inherently resilient to node failure, which is not available in the normal routing protocols. Since the capture effect is universal, as long as the network is connected, the sudden failure of a few nodes will not cause failure in path identification. The path identification is also lightweight and fast. For both testbeds, the size of the path-ident packet is rather small, of 16 bytes, which has both advantages of short radio-on time and high reception rate. However, the results also show the discrepancy between the experimentation and theory. In the testbed

evaluation, I find that the distribution of the identified paths is concentrated on a few short paths (the most common 3 paths together have probability $> 90\%$), but has a long tail (29 and 12 paths are identified for both flows respectively). I argue that the reason that many paths of different length are identified is mainly due to the time variation of the wireless channel. Yet it is favorable as it provides me with the chance to attain high reliability by combining multiple good paths. Furthermore, I confirm that despite the increase of path length, the path identification remains effective when I use a smaller transmission power.

WSNShape Protocol After I have identified the reliable paths, I am ready to utilize them to improve the QoS. The WSNShape protocol takes a parameter of *path count* C, which is the number of different paths I combine to form a stripe. C can be ∞, meaning that all paths should be combined. The WSNShape protocol performs the following steps for a flow continuously:

1. Path Identification. As described above, paths are continuously identified in the control slots.

2. Path Combination. At the destination node I use a sliding window of size M, holding the most recent M paths identified (the default value is $M = 100$, keeping a history of 10 to 20 min). After a new path is put into the sliding window, I perform a sort on the paths in the decreasing order of path frequency. Then I combine the C most common paths to form a stripe [9].

3. Stripe Activation. If the nodes in the stripe have been changed, the destination node floods it in the form of bit map in the next control slot of the opposite flow. To guarantee a high probability of reception, the packet is flooded three times. When a node receives the stripe, it checks whether it is in the stripe or not. Based on that, it activates or deactivates itself (sleeps) in the future data slots for the flow.

Since the WSN nodes are generally resource constrained devices with limited RAM and computational capability, I need to optimize the data structure of the sliding window. Although a relatively large number of path samples (up to 100) are preserved, normally the number of different paths is an order of magnitude less. Therefore, I use a linked list to keep these samples. A node in the linked list consists of a path and the number of occurrence of it. The data structure is efficient in terms of both storage and computation. In the next section, I will evaluate how the WSNShape performs compared to the baseline Glossy.

5.2.5 *Performance Comparison of Different Sparkle Operation Modes*

A Sparkle operation mode is a given combination of a transmission power and a topology control. As shown by extensive evaluation in this section, different operation modes have different performance trade-offs in reliability, latency and energy consumption. Eight operation modes are investigated, which have different topology control or transmission power as listed in Tab. 5.2. The evaluation gives insight into the performance of different operation modes, providing fundamentals for the design of an adaptive scheme.

9 In my implementation, the statistics starts when 10 paths are identified.

Operation Mode	Topology Control	Transmission Power
BL-HI	All nodes active	0dBm
BL-LO	All nodes active	−15dBm
NS-1	WSNShape with path count $C = 1$	0dBm
NS-2	WSNShape with path count $C = 2$	0dBm
NS-3	WSNShape with path count $C = 3$	0dBm
NS-4	WSNShape with path count $C = 4$	0dBm
NS-5	WSNShape with path count $C = 5$	0dBm
NS-ALL	WSNShape with path count $C = \infty$	0dBm

Table 5.2: Eight Sparkle operation modes

5.2.5.1 *Evaluation Setup*

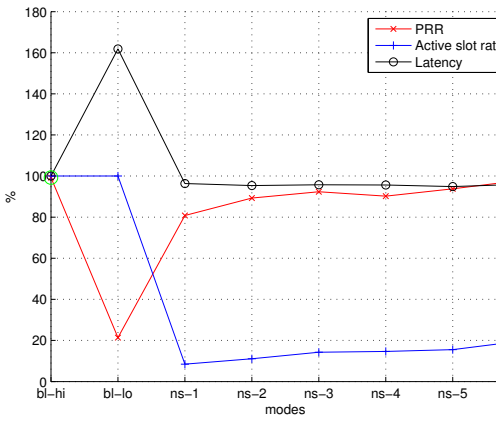
In the evaluation, the Sparkle frame has a period of 1s. In each one-hour run, I evaluate 6 flows simultaneously. The 6 flows are 3 pairs of opposite flows (e.g., $a \leftrightarrow b$ is a pair). Each flow needs to transmit a packet per second (corresponding to a control system with period of 1s). Thus, the frame is composed of 8 slots — one sync slot, 6 data slots, and one control slot which is circularly used by each flow to identify paths and to broadcast the stripe of WSNShape for its opposite flow. To save energy, all packets except the stripe broadcast are transmitted only once. The stripe broadcast is transmitted 3 times. The stripe for a given path count is broadcast whenever it is updated. The data packet has a length of 126 bytes. Furthermore, in each slot, a node turns off the radio when it has transmitted for the given number of times (once or thrice) or it has been on for 40ms. The same as before, to fairly compare different modes, the network circularly runs in each mode for 1s. This is controlled by the *sync-seq*, a counter contained in the sync packet, incremented after each frame. After the network is bootstrapped, each node should share the same sync-seq. It also controls which flow should use a certain control slot. The program is implemented on the Contiki OS [DGV04].

I evaluate two types of flows: 1) long flow, with end nodes far apart and 2) unreliable flow, where the flow itself or the opposite flow has low reliability ($< 90\%$) in the default BL-HI mode (same as the default Glossy flooding). [10] These flows represent the worst evaluation scenario since path identification for them should be relatively ineffective. However, for all flows, the path identification is successful. The long flow set includes 6 flows (3 pairs) for the Piloty and Arena testbeds each. The unreliable flow set includes 22 flows (11 pairs) for the Piloty testbed and 24 flows (12 pairs) for the Arena testbed.

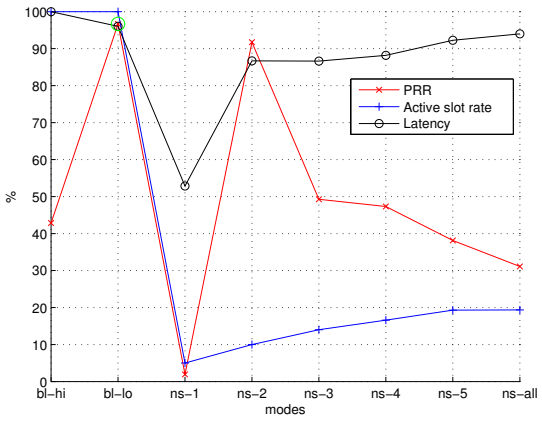
5.2.5.2 *Performance Comparison*

The QoS metrics of a number of typical flows are depicted in Fig. 5.16 and the average values of the QoS metrics over all flows are shown in Tab. 5.3. I focus on the trade-off of 3 metrics: 1) PRR, the end-to-end reliability of packet delivery of a flow, 2) Action

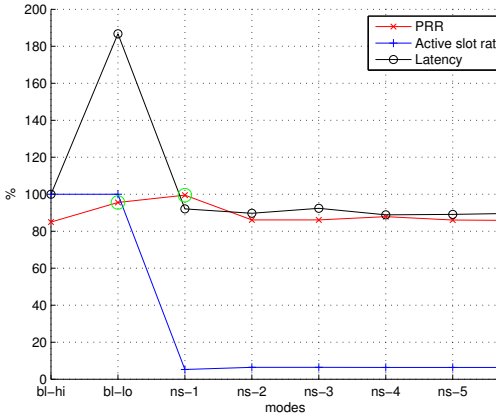
10 These pairs of flows are identified by the experiments in Sec. 5.2.4.1 for evaluating the effects of transmission power.



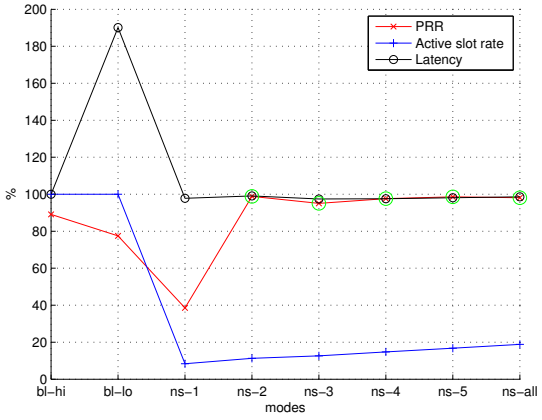
(a) flow $n_{29} \rightarrow n_{60}$, best mode: BL-HI



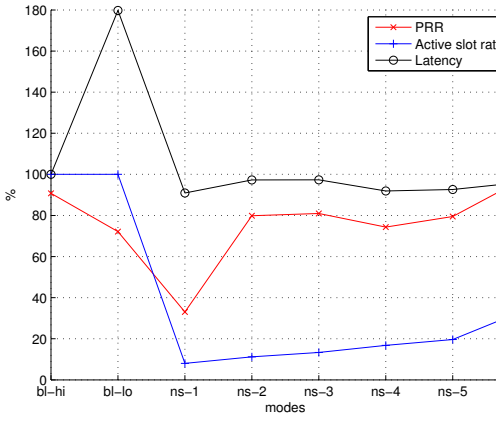
(b) flow $n_{1044} \rightarrow n_{1001}$, best mode: BL-LO



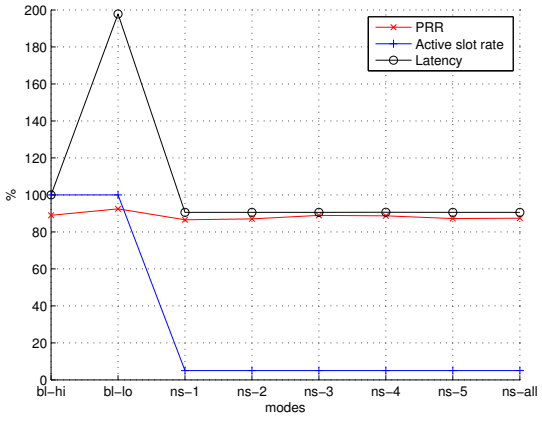
(c) flow $n_{1015} \rightarrow n_{1032}$, best mode: NS-1



(d) flow $n_{60} \rightarrow n_{33}$, best mode: NS-2



(e) flow $n_{60} \rightarrow n_{29}$, best mode: NS-ALL



(f) flow $n_{1024} \rightarrow n_{1048}$, PRRs $< 95\%$.

Figure 5.16: Some typical results from the evaluation of Sparkle modes. The points highlighted with a green circle have good reliability (PRR $\geqslant 95\%$). Latencies are normalized to that of BL-HI.

Table 5.3: The statistical QoS results of various Sparkle modes. The column "adaptive" refers to the combination of BL-HI, BL-LO, NS-2 and NS-ALL. All metrics are in percentage. Good PRR means PRR $\geqslant$ 95%.

	BL-HI	BL-LO	NS-1	NS-2	NS-3	NS-4	NS-5	NS-ALL	adaptive
Good PRR Rate	58.62	55.17	56.90	68.97	77.59	81.03	81.03	86.21	98.28
Normalized Latency	100.00	169.26	91.81	92.91	93.13	92.79	93.06	93.93	-
Active Slot Rate	100.00	100.00	6.66	9.14	11.62	13.28	14.58	18.23	-
Best Reliability Rate	18.97	17.24	24.14	25.86	27.59	43.10	34.48	36.21	75.86

Slot Rate (ASR), the number of active data slots over the number of all data slots for a flow and 3) normalized latency, the average end-to-end latency normalized over the value of the mode BL-HI. The active slot rate should be the same as the average percentage of active nodes, which is roughly proportional to the energy consumption.

Reliability In average, the reliability of all WSNShape modes except NS-1 is better than that of the baseline modes BL-HI and BL-LO (Tab. 5.3). The mode NS-ALL is generally the best. The situation that NS-1 is significantly worse than the other WSNShape modes evidences the advantage of constructive interference of multiple transmitters in boosting reliability. It shows that the concurrent transmission based network with only a few concurrent transmitters is more reliable than the traditional network based on single-path routing, even if the path is reliable. On the other hand, if the number of concurrent transmitters is very high (mode BL-HI), the reliability may decrease. Furthermore, there is at least one flow for which a certain mode is better than all the others (Fig. 5.16). Thus, there is no winner in all cases and the relative reliability among various modes can be arbitrary for individual cases. However, statistically, I can have the following reliability model:

$$R(\text{NS-ALL}) > \cdots > R(\text{NS-}i) > R(\text{NS-}j) > \cdots > R(\text{NS-2}) > R(\text{BL-HI}) > R(\text{NS-1}) > R(\text{BL-LO}) \tag{5.11}$$

where $R(\cdot)$ is the reliability of a mode, and $i = j + 1$.

Moreover, if I can always choose the best mode among BL-HI, BL-LO, NS-2 and NS-ALL, over 98% of the cases, I can obtain good reliability ($\geqslant$ 95%, sufficient for most control systems). I include BL-HI and BL-LO because they are same as Glossy, but running at different powers. I include NS-2 and NS-ALL because the first is the concurrent transmission mode with the highest energy efficiency while the second is the concurrent transmission mode with the highest reliability. The only case that good reliability is unattained is the pathological case shown in Fig. 5.16(f) where only one path can be identified for the flow and none of the modes reaches good reliability. This motivates me to design an adaptive scheme which can adaptively choose among the four modes.

Additionally, I list in Tab. 5.3 the Best Reliability Rate, which is the percentage of times that each mode achieves the highest reliability of all evaluated modes. The values do not sum to 100% because multiple modes may achieve the same highest reliability. The corresponding value under "adaptive" denotes the percentage of times that any of the four modes — BL-HI, BL-LO, NS-2 and NS-ALL achieves the best reliability.

End-to-end Latency As listed in Tab. 5.3, the average latency of the WSNShape modes are 6% to 8% shorter than that of the BL-HI, and generally the latency increases slightly with the path count C. This shows another advantage of limited concurrent transmissions. The moderate number of concurrent transmitters increases SNR and thus PRR in comparison to the large number of concurrent transmitters in the mode BL-HI. Therefore, statistically fewer rounds of retransmissions are needed till a packet successfully reaches the destination. The latency of the BL-LO mode is about 69% longer than that of the BL-HI mode, because the lower transmission power increases the hop count. In general, the end-to-end latency is near optimal. The largest average latency and hop count of a flow under the low transmission power of $-15\mathrm{dBm}$ are $33.7\mathrm{ms}$ and 7.0, respectively. The values under the high transmission power of $0\mathrm{dBm}$ are $18.9\mathrm{ms}$ and 3.7, respectively. The latencies are very small considering the large packet size of 126 bytes whose transmission takes more than $4\mathrm{ms}$ per hop. In addition, Sparkle can provide the hard deadline guarantee by turning off the radio after the slot duration has finished (slot duration $= 40\mathrm{ms}$ in my implementation).

Energy Consumption The actual energy consumption should be roughly proportional to the ASR. Obviously ASR = 100% in the baseline modes because all nodes are active. The various WSNShape modes NS-1 to NS-ALL save as much as 93% to 82% of energy compared to the baseline modes. This is due to the large amount of inactive modes. Intuitively, the saving decreases with the path count C. Although the ASR value of the two baseline schemes is 100% in both cases, I expect that BL-LO consumes more energy than BL-HI, since the former has a much longer latency which increases the radio-on time significantly. This more than compensates the slight saving brought by the low transmission power. In summary, I can give an energy consumption model:

$$E(\text{BL-LO}) > E(\text{BL-HI}) \gg E(\text{NS-ALL}) > \cdots > E(\text{NS-}i) > E(\text{NS-}j) > \cdots > E(\text{NS-1}) \quad (5.12)$$

where $E(\cdot)$ is the energy consumption of a mode, and $i = j + 1$.

Summary In general, the reliabilities of the WSNShape modes improve with the path count C. The trade-off is that the latencies and energy consumptions (ASR) of them increase with C (Tab. 5.3). Compared to the Glossy protocol (with different transmission powers), WSNShape with $C \geqslant 2$ brings improvement in reliability, latency and energy consumption simultaneously. The energy saving is significant, over 80%. The improvement in latency is slight, only a few percent. Regarding reliability, NS-ALL is generally the best mode. But the relative reliabilities of different modes could be arbitrary for a flow. However, if I can adaptively choose the most reliable mode among BL-HI, BL-LO, NS-2 and NS-ALL, far better performance can be achieved than sticking to any specific mode, which is the main topic of the next section.

5.2.6 *PRRTrack: Adaptively Minimizing Energy Consumption while Meeting Reliability Requirement*

A useful control system must be stable and have a satisfactory performance, which normally requires that each flow has latency below and reliability above preset values [ZBP01]. In this section, I design and evaluate an automatic scheme, PRRTrack, a component of Sparkle that adaptively switches between different operation modes, with the goal of minimizing energy consumption while meeting the reliability re-

quirement. In case the reliability requirement cannot be satisfied by any of the modes, PRRTrack achieves the best-effort performance by keeping a flow operate in the most reliable mode for most of the time. The testbed evaluation shows that PRRTrack effectively achieves its design goal together with the advantage of improved latency.

5.2.6.1 *The Design of PRRTrack*

The main idea of PRRTrack is simple: if the current mode satisfies the reliability requirement, it tries to find a more energy-efficient one, otherwise it tries to find one that satisfies the reliability requirement. Given the model of relative energy efficiency of the various operation modes, the process to find a more energy-efficient mode is straightforward. But on the other hand, since no deterministic model of the relative reliability is available, the process to find a mode satisfying the reliability requirement is based on trial-and-error.

The control logic of PRRTrack is realized at the destination node of a flow. It performs two activities: first, it maintains the recently identified 100 paths for WSNShape; second, it keeps track of the *current PRR* by calculating the reception rate of the recent 100 data packets of a flow. Also, in the manner of feedback control, it gives proper commands of mode switch based on the difference between the current PRR and the reliability set-point.

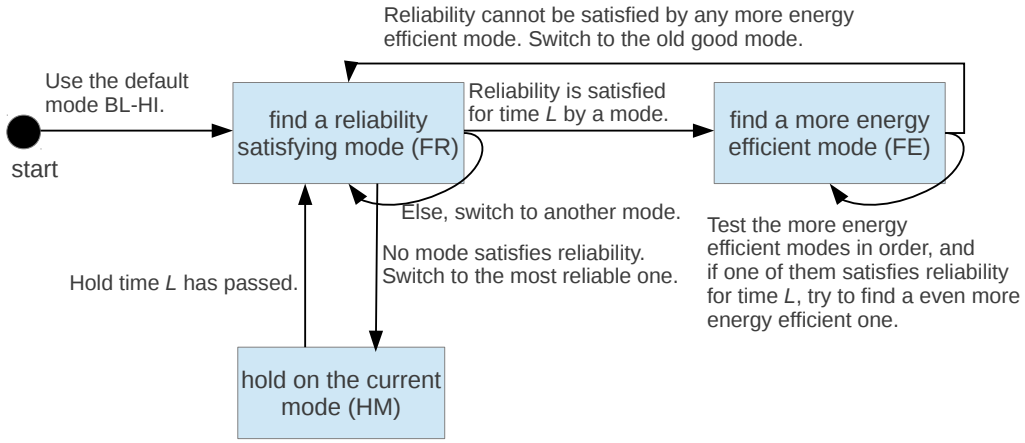


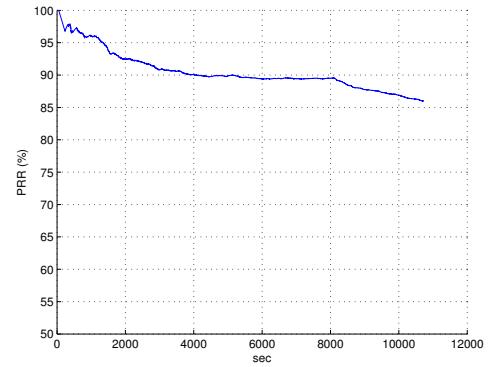
Figure 5.17: The operation mode switch process of PRRTrack.

The operation mode switch process of PRRTrack is illustrated in Fig. 5.17. I only switch among the four modes BL-HI, BL-LO, NS-2 and NS-ALL, since I prefer to minimize mode switches. I revise my energy model from (5.12):

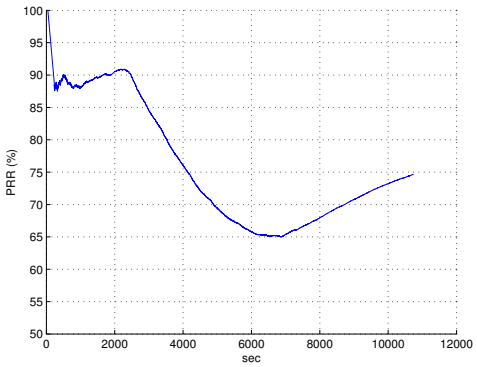
$$E(\text{BL-LO}) = E(\text{BL-HI}) > E(\text{NS-ALL}) > E(\text{NS-2}) \tag{5.13}$$

5.2.6.2 *Implementation*

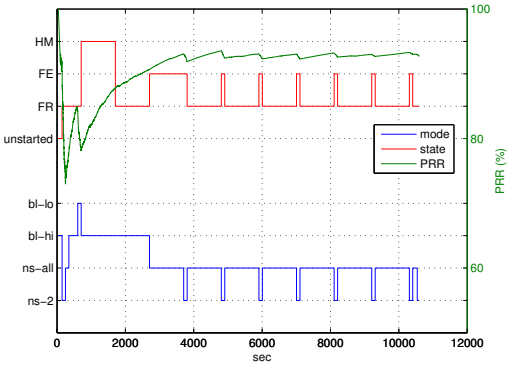
The implementation details of PRRTrack are as follows. The control slots are used for path identification and mode switch commands. The path-ident packets are sent by the source node once every 10s if there is no pending mode switch command, which is sent by the destination node whenever necessary. Included in the mode switch command is the stripe for WSNShape, if the mode is a WSNShape one. Since the mode switch features trial-and-error, in my implementation it may lead to temporary



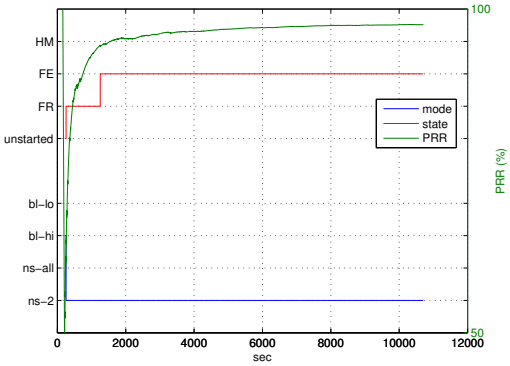
(a) flow $n_{26} \rightarrow n_{63}$, fixed mode BL-HI



(b) flow $n_{16} \rightarrow n_{61}$, fixed mode BL-HI



(c) flow $n_{26} \rightarrow n_{63}$, PRRTrack



(d) flow $n_{16} \rightarrow n_{61}$, PRRTrack

Figure 5.18: The reliability of PRRTrack vs fixed mode. Reliability requirement $R = 90\%$. HM stands for "hold on the current mode"; FE stands for "find a more energy efficient mode" and FR stands for "find a mode satisfying the reliability requirement".

PRR decrease when I probe a new mode. In the chosen configuration, this may cause decreased performance of about 100s (the test duration, e.g., Fig. 5.18(c)). If the control system is sensitive to that, I could implement the PRRTrack more conservatively by probing a mode in the control slots before actually switching the data slots to that mode. This will give better reliability performance at the cost of more energy consumption due to the higher overhead of control slots and slower mode switch reaction. Whenever I switch to a different mode, I flush the sliding window for PRR re-calculation.

5.2.6.3 *Evaluation*

For the evaluation of PRRTrack, I compare the performance of Sparkle with PRRTrack to that of Sparkle with the fixed mode BL-HI (same as the default Glossy flooding). In each round of the evaluation, I run both programs for 3 hours each. Similar to Sec. 5.2.5.1, 3 pairs of opposite flows are evaluated in each round. For each of the Piloty and Arena testbeds, I evaluate 12 random pairs of opposite flows.

The evaluation results of two representative flows in the Piloty testbed are shown in Fig. 5.18. I show the averaged PRR from the program start. From Figures 5.18a and

5.18b I observe that if I stick to the BL-HI mode, neither of the two flows can satisfy the reliability requirement of PRR $\geqslant R$ with $R = 90\%$. Furthermore, the PRR values are not stable. There are long periods in which the PRR goes up or down steadily. From Figures 5.18c and 5.18d I see that if PRRTrack is applied, the reliability requirement can be satisfied and the PRR values are much more stable.

In the case of Fig. 5.18(c), I observe that after the PRRTrack starts, I transition immediately into the state FR (FR stands for "find a reliability satisfying mode"). Then I test the modes NS-2, NS-ALL, BL-HI and BL-LO one by one, in the decreasing order of energy efficiency, to find a reliable one, but none of them satisfies PRR $\geqslant 90\%$ for 1000s ($L = 1000$s). Therefore, I switch to the most reliable mode at that time, BL-HI, and hold on it for 1000s. After the hold time, I transition back to the state FR. But now BL-HI can meet R for over 1000s, therefore I later transition to the state FE (FE stands for "find a more energy efficient mode"), to find a more energy-efficient mode. Now I land in the mode NS-ALL, which is not only more energy-efficient, but also reliable enough. But from time to time (after about every 1000 sec), I try the more efficient NS-2 for 100s, to find a potentially reliable and more energy-efficient mode. However, until the end of the program, this probing is unsuccessful. The situation in Fig. 5.18(d) is simpler. After I switch to the mode NS-2, the reliability requirement can always be satisfied, therefore, I stay in the mode as it is already the most energy-efficient one.

Table 5.4: The energy consumption and latency of PRRTrack vs. those of the fixed mode. Energy_d is the energy consumption of data slots. Energy_c is that of control slots.

	Fixed mode BL-HI (3 hours)			PRRTrack (3 hours)		
	$\text{Energy}_d(J)$	$\text{Energy}_c(J)$	Latency(ms)	$\text{Energy}_d(J)$	$\text{Energy}_c(J)$	Latency(ms)
flow $n_{26} \leftrightarrow n_{63}$	1008.27	0	18.09	196.21	24.11	16.15
flow $n_{16} \leftrightarrow n_{61}$	1009.68	0	16.11	109.21	23.17	14.45

To measure the energy consumption of the radio component, which accounts for the predominant part of energy consumption of my system, I use Energest, a software-based method for energy measurement provided in Contiki [DÖTH07]. For higher precision, I consider the different current consumptions of listen mode and transmit mode with various transmission powers provided by the CC2420 datasheet [CC213]. As listed in Tab. 5.4, the energy saving of the PRRTrack is huge. For two pairs of flows $n_{26} \leftrightarrow n_{63}$ and $n_{16} \leftrightarrow n_{61}$, including the control overhead (that of control slots), it uses only 22% and 13% of that of the fixed mode BL-HI, respectively [11]. The control overhead amounts for about 1/5 of the energy consumption. Additionally, PRRTrack also improves the average end-to-end latency by about 10%. Over all 24 pairs of flows, the average energy saving is 84% and the latency improvement is 5%.

5.2.7 *Conclusion*

I have presented Sparkle, a communication network for periodic multi-loop control systems with high packet reliability, very low energy consumption, as well as

11 I look at the energy consumption of a pair of opposite flows together instead of separately because to achieve reliable transmission on one flow, I need the cooperation of some control slots of its opposite flow.

near-optimal and deterministic communication latency. To my knowledge, this is the first control network based on concurrent transmission. Sparkle has a flexible architecture that supports arbitrary and independent QoS control mechanisms for all communication flows. The novel WSNShape is a topology control technique based on the capture effect. It leads to huge saving of energy consumption as well as to high probability of improvement in reliability and latency, compared to the Glossy protocol. By combining different levels of WSNShape and transmission power, I derive various operation modes featuring different energy and performance characteristics. Then I design a control scheme PRRTrack, that can adaptively switch between these operation modes. Through extensive evaluation on real-world testbeds, I confirm that my scheme PRRTrack satisfies the design goal of preset reliability while in average massively reducing the energy consumption by 84%. In addition, it also reduces the latency by 5%.

5.3 RIPPLE: HIGH-THROUGHPUT, RELIABLE AND ENERGY EFFICIENT NETWORK FLOODING IN WIRELESS SENSOR NETWORKS

5.3.1 *Introduction*

The Glossy protocol [FZTS11] is multi-hop network flooding protocol of high reliability, low latency and low energy consumption. Yet it can be further improved in two aspects: (1) the potential throughput of the IEEE 802.15.4 standard is not fully utilized; (2) it is not robust to interference. Since it only uses one physical channel, the network is susceptible to interference on that channel. High-throughput and robustness to interference are indispensable for some applications, e.g., one-to-all multimedia broadcast. This thesis proposes a new network broadcast protocol, called Ripple, which improves Glossy in these two aspects. Additionally, it supports a flexible configuration of throughput and packet reliability, making itself suitable for a large spectrum of applications with various QoS requirements. Furthermore, it is far more energy efficient than Glossy, and achieves higher packet reliability than Glossy through forward error correction.

Ripple has two novel features over Glossy: *pipeline transmission* and *erasure coding*. Instead of flooding one packet over the network in each *round* as Glossy does, the pipeline transmission floods a number of distinct packets on multiple channels in each round. In a Glossy round, a node may receive the same packet multiple times, where the redundant receptions do not contribute to throughput; while in a Ripple round, a node receives different packets and each contributes to the throughput. By using only two channels, I can practically eliminate the mutual interference among different packets being transmitted simultaneously on the same channel. Different from the traditional node-based channel assignment [RCBG10, DCL13b], Ripple uses a novel packet-based channel assignment. This saves the need for a channel assignment phase, which requires a specialized protocol and carries extra overhead as in, for instance, the Splash protocol [DCL13b]. Although it significantly increases the throughput, the pipeline transmission also decreases the chances to receive a certain packet. To compensate for that, I apply the optimal Reed-Solomon (RS) erasure code [Riz97], which encodes k packets into n packets for arbitrary $n \geqslant k$. If any k of the n packets are received, the original packets can be decoded and restored. Moreover, using

multiple channels helps to improve the robustness to interference in the 2.4 GHz ISM band. Even if part of the channels are fully jammed, erasure coding can guarantee good reception of original messages as long as the packet reception rate is higher than the code rate. Although the aim of Ripple is high throughput and high reliability network broadcast, rather than fully reliable data dissemination, it still makes sense to compare Ripple with protocols for reliable data dissemination, since Ripple achieves over 99% reliability when applying forward error correction. Compared to the state-of-the-art data dissemination protocol Splash, Ripple obtains two to three times higher throughput.

The rest of the section is organized as follows: Sec. 5.3.2 discusses the related work. Sec. 5.3.3 elaborates on the design of Ripple. Sec. 5.3.4 analyzes the theoretical throughput gain achievable by Ripple, compared to Glossy. Sec. 5.3.5 performs some preliminary experiments on the implementation choices and the guidelines for choosing Ripple parameters. Sec. 5.3.6 shows the evaluation results of Ripple on three testbeds. Finally, Sec. 5.3.7 concludes the section.

5.3.2 *Related Work*

Ripple draws primarily on the seminal work of Ferrari et al., Glossy [FZTS11]. Glossy is a network flooding protocol exploiting the physical layer feature of constructive interference. It achieves low latency, high reliability and time synchronization of µs accuracy simultaneously. Ripple enhances Glossy by adding pipeline transmissions on multiple channels, which improves throughput and energy efficiency, as well as incorporating forward error correction, which improves packet reliability.

The general idea of employing pipeline transmission on multiple channels to improve throughput is not new. By doing so, PIP [RCBG10] increased the end-to-end throughput of a multi-hop line topology and Splash [DCL13b] enlarged the network throughput of CI-based data dissemination. The Ripple protocol differs from these two protocols in that it assigns a channel to a packet instead of to a node. Therefore, PIP and Splash both require a one-shot or continuous phase that assigns channels to nodes. For instance, Splash uses the CTP protocol to derive the level of a node, and assigns a fixed channel to each level before performing pipeline transmissions. This incurs extra overhead, which is not described in [DCL13b]. In contrast, Ripple's packet based channel assignment does not call for such a phase. As long as the nodes are time synchronized, they can switch channels correctly.[12] The result is that Ripple is more adaptive to topology change, more compatible with the routing-free nature of CI-based protocols, and has two to three times larger throughput than that of the state-of-the-art data dissemination protocol Splash (10 Kbits/s) even after error correction. Thus, it transitively outperforms the popular data dissemination protocol Deluge T2 [HC04].

Packet corruption often happens in concurrent transmissions and forward error correction can help to improve the packet reliability significantly [PJJ+14]. My work is the first to apply the efficient Reed-Solomon erasure code [Riz97] to concurrent transmissions in WSN. The RS code can be tuned according to communication reliability and can thus satisfy any requirement concerning this issue.

12 I perform continuous hop estimation at each node to further improve the performance, but it is done at no extra cost. This process is described in Sec. 5.3.3.1

5.3.3 *The Design of Ripple*

In each Glossy round, one packet is flooded from the *initiator node* to the whole network, which is similar to a circular wave expanding on a water surface. This feature limits the throughput, especially when the hop-diameter of the network is large. Ripple addresses this problem by flooding multiple packets in each round. Conceptually, it resembles a train of circular waves expanding simultaneously.

If multiple packet floodings all use a single channel, the simultaneous transmissions of different packets will interfere with each other. To avoid it, Ripple uses multiple channels cyclically. When two packets of the same channel are simultaneously active, there is enough distance between them to prevent mutual interference.

5.3.3.1 *Protocol Design*

I elaborate on the design of Ripple in this part. First, I describe the frame structure. Next I discuss the slot scheduling of a node. Then I describe the hop estimation that avoids cascading packet loss, and finally, I describe the RS erasure code.

FRAME STRUCTURE A Ripple frame consists of one *sync round* and a number of *data rounds*. In the sync round, an *authority node* [13] performs a Glossy flooding of a very short sync packet so that all nodes get synchronized with it. Theoretically, the sync round is not necessary, because I could reuse the data packets for synchronization at no extra cost. However, I decide to keep it since I need reliable and highly accurate synchronization for the correct operation of the pipeline transmissions. The flooding of very short sync packets meets my requirement, incurring only a negligible cost of frame length, but providing a dependable basis for the correct operation of Ripple. The data rounds are used for one-to-all data broadcast. In each data round, a batch of data packets are flooded in pipeline.

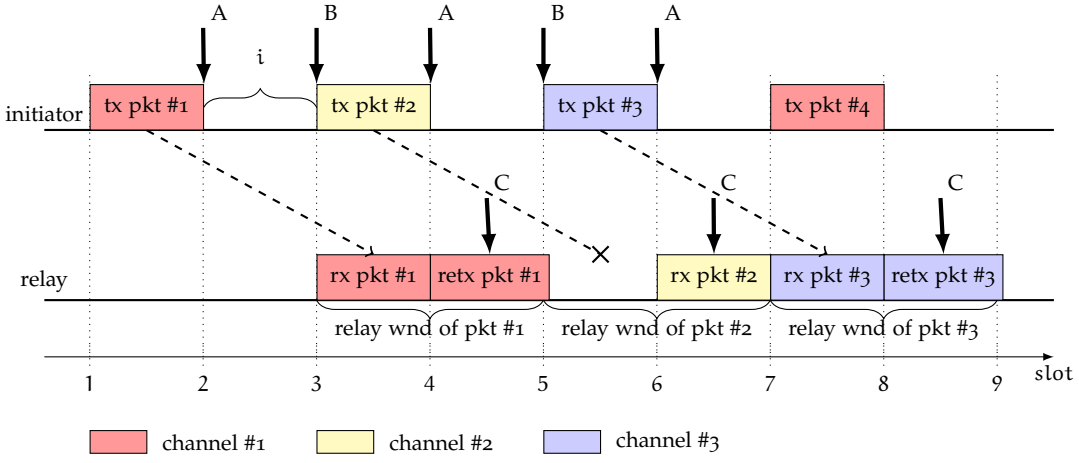


Figure 5.19: An example of the slot scheduling in a data round. The distance between the initiator node and the relay node is 3 hops. 3 channels are used and the transmission interval is $i = 1$.

13 The authority node can be different to the initiator node.

SLOT SCHEDULING IN DATA ROUNDS In the network, the initiator node broadcasts the data packets on the available channels sequentially and cyclically. All the other nodes are *relay nodes* and retransmit each distinct data packet received just once. The slot scheduling is essential for the pipeline transmissions on multiple channels. I illustrate this concept in Fig. 5.19. Note, a *slot* refers to the communication duration of one data packet.

The initiator broadcasts a batch of packets circularly on the available channels every $i+1$ slots in a data round, where i is called *transmission interval* and defines how fast the initiator performs pipeline transmissions ($i \geqslant 1$). Each relay continuously estimates the *hop distance* to the initiator, which is described in the next subsection. From the estimated hop distance, a relay determines a *relay window* of $i+1$ slots for each packet, corresponding to $i+1$ reception chances (Fig. 5.19). To receive a packet, the relay switches to the channel of the packet before the start of the *relay window*. If the packet is received and there are remaining slots in the relay window, the node relays it immediately on the same channel in the next slot, e.g., pkt #1 and pkt #3 in Fig. 5.19. The case that a packet is received but there is no more slot for retransmission is illustrated by pkt #2. At the end of the relay window, irrespective of the result of the packet reception, the node switches immediately to the channel of the next packet, preparing for its reception.

The initiator turns off the radio after a packet transmission to save power, then it writes the frequency of the next channel to the radio chip (at instants labeled with A in Fig. 5.19). This helps to save the turn-around time since at the next turn-on instant, it will directly switch to the correct channel. The turn-on of the radio for packet transmission (at instants B) is controlled by a timer with an interval of $i+1$ slots. A timer of the same interval is also needed at a relay. When it is fired (at instants C, the middle point of the last reception chance), the relay checks whether a packet reception or transmission is ongoing. If this is the case, it switches to the channel of the next packet as soon as the current reception or transmission finishes. Otherwise, it immediately switches channel, which helps to avoid the cascade effect of a packet loss.

HOP ESTIMATION In order to receive as many packets as possible, a relay node needs a good estimation of the hop distance to the initiator. The logic underpinning hop estimation is shown in Alg. 13.

Algorithm 13: Estimation of the hop distance to the initiator.

Data: i: transmission interval

Initialize the estimated hop distance d_e to a value larger than the network diameter;
for *Each data round* **do**
    Cyclically shift the channels once;
    Start the batch transmissions;
    if *Receive the first packet of the batch* ***and*** *the measured distance* $d < d_e + i$ **then**
        Add d to the FIFO set D;
    if *Receive the last packet of the batch* ***and*** *the first packet is lost* ***and*** *the measured distance* $d \geqslant d_e + i$ **then**
        Add d to the FIFO set D;
    Update the estimated hop distance with $d_e = \arg\max_k \#(\{j : j \in D, k \leqslant j < k + i\})$;

At the start of the Ripple protocol, each relay node initializes the estimated hop distance d_e to a sufficiently large value (Line 1). Then, at the end of each data round, I update the estimation based on the received packets. The basic idea is that, in order to be adaptive to the change of the topology, I collect the measured distances in a FIFO set of fixed size, and then, based on the distribution of collected distances, I give an appropriate estimate (Line 9). To get an unbiased estimation, I only look at the samples of the first and the last packet of each batch. The reason is that if the current estimated hop is too small, then the last packet will probably give a sample of larger distance because it is only loosely restricted by the end time of the data round. On the contrary, if the current estimate is too big, then the first packet will probably give a sample of smaller value as all nodes start the data round simultaneously. The packets in the middle are restricted in both ends by the slot scheduling and thus, they are unsuitable for estimation. I cyclically shift the channels once at the beginning of each data round (Line 3) since the estimated hop distances also depend on the channel, and therefore, I want to give each channel the same chance to be sampled. The algorithm is very effective, and it even gives better performance than the estimation based on the sync rounds that uses the same packet length as that of the data packets. But long sync rounds would waste too much time.

REED-SOLOMON ERASURE CODE I recapitulate the RS erasure code used in Ripple. For more detail, please refer to [Riz97]. The IEEE 802.15.4 standard specifies a 16-bit Cyclic Redundancy Check (CRC) for packet verification [Soco6]. Hence, I can model the communication channel as erasure, i.e., the packets passing the CRC check are correctly received while the others are completely lost.

The RS erasure code is a (n, k) linear block code, where k packets are encoded at the initiator into n packets. If k or more packets are received, a receiver can recover the k original packets. The RS code is based on the Vandermonde matrix. Specifically, a (n, k) linear code can be represented as

$$y = Gx \tag{5.14}$$

where x is a $k \times 1$ source data, G is a $n \times k$ matrix and y is the $n \times 1$ encoded data. In order to represent the components of x and y in the same number of bits, the arithmetics are performed in the Galois field $GF(p^s)$, with p prime and $s > 1$. In addition, I use systematic RS code, meaning G consists of two parts, the $k \times k$ identity matrix I_k and the $(n-k) \times k$ Vandermonde matrix V.

$$G = \begin{pmatrix} I_k \\ V \end{pmatrix} \tag{5.15}$$

V can be represented in the form of

$$V = \begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{k-1} \\ 1 & \alpha_2 & \alpha_2^2 & \cdots & \alpha_2^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_{n-k} & \alpha_{n-k}^2 & \cdots & \alpha_{n-k}^{k-1} \end{pmatrix} \tag{5.16}$$

where $\alpha_1, \ldots, \alpha_{n-k}$ are nonzero elements of $GF(p^s)$. The reason I can recover the source packets from any k received packets is that any k rows from G are linearly

independent. According to the properties of a Galois field, $k-1 < p^s-1$ and $n-k < p^s$, I have $k_{max} = p^s-1$ and $n_{max} = 2(p^s-1)$. I choose $p = 2$ and $s = 8$, which means I perform GF arithmetics on bytes. This has as advantages that the computation time is faster and that the overhead is acceptable in terms of the look-up tables for speeding up the arithmetics. For this choice of p and s, I have $k_{max} = 255$ and $n_{max} = 510$, i.e., I can encode at most 255 source packets into 510 packets in a batch. Any combination of n and k with $n \leqslant n_{max}$ and $k \leqslant k_{max}$ is allowed. Considering that normal sensor nodes have only a dozen kilobytes of RAM and that the maximum IEEE 802.15.4 packet size is 127 bytes, $p = 2$ and $s = 8$ can be universally used for any application.

5.3.4 *Throughput Gain in Theory*

In this section, I deduce the theoretical throughput gain of Ripple compared to Glossy under the assumption of perfect packet reliability, considering both cases with and without error coding. For both Glossy and Ripple, suppose that the throughput under perfect reliability is P, then the throughput under the reliability R will be $P' = P \cdot R$.

5.3.4.1 *Ripple without Error Coding*

I first compute the theoretical throughput of Glossy. Suppose the network diameter (the maximum hop distance from any node to the initiator node) is h. The slot time is t_{slot}, which is the sum of the transmission time of a packet and the turn-around time, i.e., the interval between the end of a packet transmission and the start of a packet relay. Then, to make sure that each node receives the packet, the round time should be at least $h \cdot t_{slot}$. Thus, the throughput of Glossy, P_{glossy}, is at most

$$P_{glossy} = \frac{1}{h} \text{ packets/slot} \tag{5.17}$$

In contrast, Ripple floods multiple packets in a round. Suppose that a batch of k packets are flooded in each round with $k > 1$, and a packet is flooded every $i+1$ slots. Then, the initiator takes at least $k(i+1)t_{slot}$ for the communication. The same occurs with each relay, although the communication time of a relay node may be delayed with respect to that of the initiator (see Fig. 5.19). Thus, the throughput of Ripple, P_{ripple}, is at most

$$P_{ripple} = \frac{k}{k(i+1)} = \frac{1}{i+1} \text{ packets/slot} \tag{5.18}$$

The gain in throughput of Ripple compared to that of Glossy is

$$G = \frac{P_{ripple}}{P_{glossy}} = \frac{h}{i+1} \tag{5.19}$$

If the pipeline transmissions are carried out at the highest speed $i = 1$, then $G = \frac{h}{2} > 1$ when $h > 2$, which is typically the case in a multi-hop network. The throughput $P_{ripple} = \frac{1}{2}$ packets/slot when $i = 1$ is intuitively the optimum value of one-to-all broadcast when the transceivers are half-duplex.

5.3.4.2 *Ripple with Error Coding*

As mentioned before, the pipeline transmissions in Ripple can potentially decrease the packet reliability compared to Glossy when the transmission interval I is small. The main reason is that there are fewer chances (slots) to receive a packet. To compensate for that, I apply the RS code. The *code rate* r, which is the ratio of the number of source packets to the number of encoded packets, is a value between 0 and 1, and should be selected according to the packet reliability of the network.

Same as before, I assume the network diameter is H and a batch of k packets are to be broadcast in each round with transmission interval I. Then $n = \frac{k}{r}$ encoded packets should be flooded in each round. Since sensor nodes are resource-constrained, I cannot ignore the time incurred by erasure coding and decoding. Following the analysis of the encoding and decoding time of the RS erasure code in [Riz97], I estimate the throughput as follows. Because I use systematic code, for a batch of k packets, I need to produce $(n-k)$ encoded packets. The time to produce a packet is proportional to the packet length l times the number of source packets k, which is reasonable as the encoding depends on each source packet. Since the slot time t_{slot} is roughly proportional to packet length l, the total encoding time can be modeled as

$$t_{enc} = c_e k(n-k)t_{slot} \tag{5.20}$$

where c_e is a constant. On the other hand, given a large packet size and a big batch size k, the time to reconstruct a missing source packet is equal to $c_d \cdot k \cdot t_{slot}$, where c_d is a constant. The values c_e and c_d will be measured for TelosB nodes in Sec. 5.3.5.2. Because the number of missing source packets m is such that $m \leqslant \min\{k, n-k\}$,[14] then the decoding t_{dec} is upper-bounded and

$$t_{dec} = c_d \cdot k \cdot m \cdot t_{slot} \leqslant c_d \cdot k \cdot \min\{k, n-k\} \cdot t_{slot} \tag{5.21}$$

The initiator and each relay take at least $n(I+1)t_{slot}$ time for the communication of a Ripple round. Therefore, by taking into account the error correction time, the throughput of Ripple becomes

$$P^{RS}_{Ripple} = \frac{k}{n(I+1) + t_{ed}} \text{ packets/slot} \tag{5.22}$$

where t_{ed} is the highest value of the encoding and decoding times, measured in number of slots and

$$t_{ed} = \max\{c_d \cdot \min\{k, n-k\}, c_e(n-k)\} \cdot k \text{ slots} \tag{5.23}$$

The throughput gain compared to Glossy is

$$G^{RS} = \frac{k \cdot H}{n(I+1) + t_{ed}} = \frac{H}{\frac{I+1}{r} + \max\{c_d \cdot \min\{1, \frac{1}{r} - 1\}, c_e(\frac{1}{r} - 1)\} \cdot k} \tag{5.24}$$

The throughput gain is thus reduced compared to the case without error correction. When $r = 1$, Eq. (5.24) reduces to Eq. (5.19), that is, the case without error correction. This result suggests that I should not use a very big batch size k, because the encoding/decoding time will almost linearly reduce the throughput for big k. The benefit

14 Otherwise the set of source packets can not be fully restored.

of error coding is that near-to error-free communication can be achieved if r is set to be smaller than the lowest flooding reliability to any node. A big advantage of Ripple (for both cases with and without error coding) is that the throughput is independent of the network size h.

5.3.5 *Preliminary Experiments*

In this section, I investigate the implementation of an accurate and precise timer for slot scheduling, the overhead of the RS erasure code, and the expected reduction on packet reliability by Ripple. These results provide reasons for my implementation choices and guidelines for the selection of Ripple parameters. The hardware and software platforms for my implementation are TelosB mote and Contiki OS. In the later experiments, I always use the maximum packet size of 127 bytes.

5.3.5.1 *Timers for Slot Scheduling*

The slot scheduling in Ripple was introduced in Sec. 5.3.3.1. Stable and precise timers are essential. Hence, I use the hardware timers of the MCU and implement the actions in the Interrupt Service Routine (ISR). I have two options in the selection of the source clock for the timers.

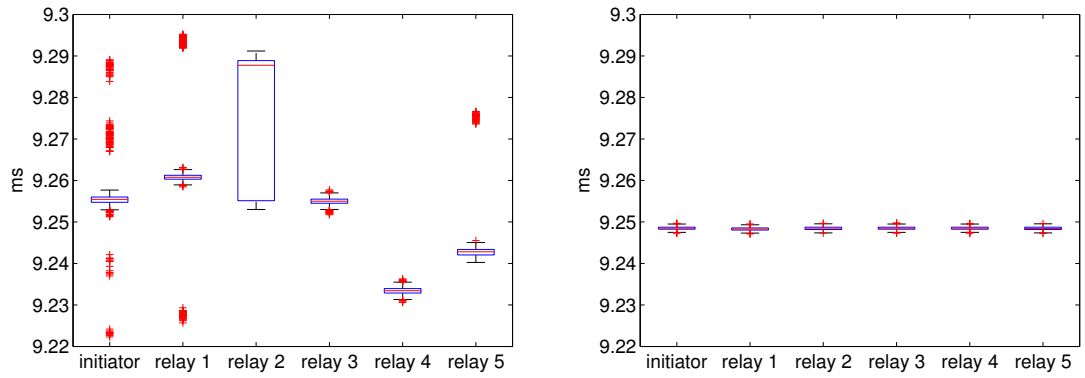
1. The timers are sourced by the DCO, which is also the main clock used by CPU. The DCO has a frequency of 4 MHz and is periodically calibrated by the crystal oscillator.

2. The timers are sourced directly by the 32 KHz crystal oscillator on board.

The DCO timer has 128x faster clock than the crystal timer, which allows for more precise control of timing. However, the frequency of DCO is notoriously instable. It drifts with temperature up to $-0.43\%/^{\circ}C$ and with voltage of power supply up to 10%/V [Tex11]. In contrast, the frequency tolerance of the crystal oscillator is only ± 20 ppm (a ppm is 0.0001%), which is far more stable [SG11]. To compensate for the instability of DCO, I calibrate it periodically to the crystal. The initiator is calibrated both at the start of each round and after each packet transmission. A relay is only calibrated at the start of each round due to the gapless transmission and reception of packets.

I evaluate the performance of these two types of timer because it is key for the correct behavior of slot scheduling. I perform Ripple with a batch size of 20 and measure the intervals between two consecutive instants labeled B at the initiator, and between two consecutive instants labeled C at the relays as illustrated in Fig. 5.19. The experiment uses six randomly selected nodes, one initiator and five relays. The timings are measured by a logic analyzer sampling at the frequency of 16 MHz on the pins set in the ISRs of the timers.

Fig. 5.20 shows that the crystal timer is far more stable than the DCO timer at individual nodes, as well as across nodes. The deviation of the intervals can be as large as 80 µs for the DCO timer. This means that if the batch size is 30, then the cumulative time offset of two nodes can be as large as 2.4 ms, which takes longer than half of the slot time of the longest packet. This may cause the relays to switch channel either too early or too late with respect to the initiator, leading to unnecessary



(a) DCO timer

(b) Crystal timer

Figure 5.20: The distribution of time intervals of two types of timers.

packet loss. The time granularity of the 32 KHz crystal (1 tick) is 30.5 µs, similar to the transmission time of a byte (32 µs), which is fine enough. Therefore, for Ripple on TelosB motes, I implement the timers for slot scheduling with the Timer_A1 in the MCU MSP430F1611 sourced by the 32 KHz on-board crystal oscillator [Tex11].

5.3.5.2 *The Runtime and Memory Overhead of the Erasure RS Code*

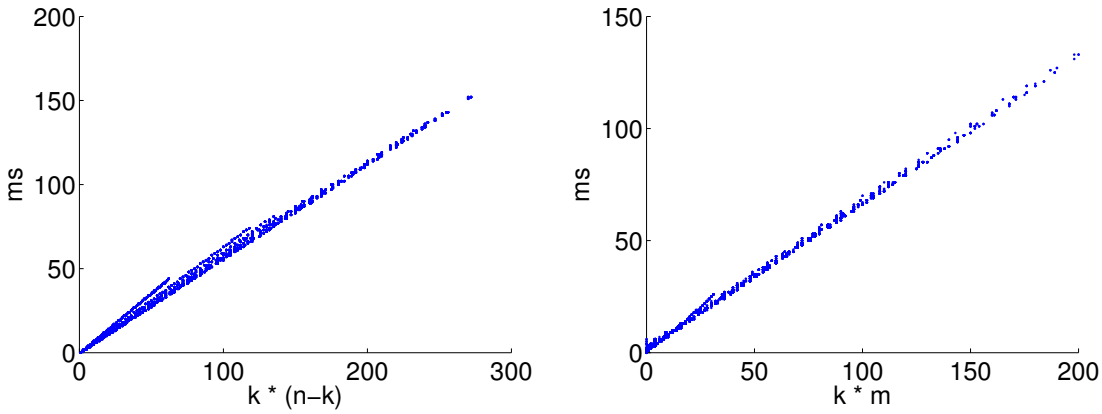
In the evaluation I increase the number of source packets k from 1 till the mote is out of memory. And for each k, I evaluate different number of encoded packets n from $(k+1)$ till out of memory. For decoding, I randomly choose k packets out of the n encoded packets as the input to the decoder, since a successful decoding needs k encoded packets, and I record m, the number of source packets to be restored.

RUNTIME OVERHEAD As shown in Fig. 5.21, the encoding and decoding times fit fairly well the theoretical models of Eq. (5.20) and (5.21). I approximate the slot time t_{slot} of l byte packets by $32 \cdot l$ µs, ignoring the time for packet preamble and software delay in Glossy. [15] A more accurate model could be derived from [YH13], but it would overcomplicate the analysis. I determine empirically the values of c_e and c_d to be $c_e = 0.14$ and $c_d = 0.17$.

MEMORY OVERHEAD The RS code encodes k packets into n packets. Since it is systematic, it takes $n \cdot l$ bytes for packet storage. For encoding, the generating matrix G takes $n \cdot k$ bytes. In addition, the lookup tables for logarithm, exponentiation and inverse operations on $GF(q)$ take $5q - 2$ bytes. Therefore, the encoder, in total, takes $n(k+l) + 1278$ bytes.

The decoder decodes n packets into k packets. Analogously, it takes $n \cdot l$ bytes for storing the packets, $n \cdot k$ bytes for the generating matrix G and $5q - 2$ bytes for the lookup tables. Additionally, it needs k^2 more bytes for the decoding matrix, which maps k received packets to k source packets. During the decoding operation, I need $m \cdot l$ bytes for the temporary storage of the m source packets to be restored. Therefore,

15 It is appropriate for the large $l = 127$ used in the experiments.



(a) Encoding (b) Decoding

Figure 5.21: Encoding and decoding times of the RS erasure code. They satisfactorily match the models of Eq. (5.20) and (5.21).

the decoder has bigger memory overhead. It takes, in total, $l(n+m)+k(n+k)+1278$ bytes, where $m = \min\{k, n-k\}$.

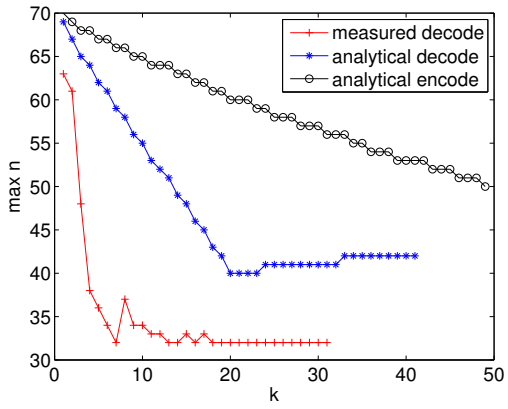


Figure 5.22: Memory overhead of the RS erasure code (maximum n for a given k without memory overflow). For analytical results, I suppose memory size is equal to 10 kilobytes.

Fig. 5.22 shows the measured maximum n for a given k for decoding before memory overflow, as well as the analytical results for encoding and decoding. The figures provide guidelines for choosing k and n. The measured decoding memory overhead is larger than the analytical result, which is mainly due to the additional overhead of Contiki OS in RAM.

5.3.5.3 *The Reduction on Packet Reliability of Ripple*

As explained before, the reduced chances for packet reception in Ripple may decrease packet reliability compared to Glossy. Here I explore the expected degree of the reduction. For that purpose, I perform Glossy floodings cyclically on the 16 channels in 2.4 GHz of IEEE 802.15.4 on the Arena testbed at TU Darmstadt [GBKVL12]. One packet is flooded in each round and each relay node only retransmits the packet once.

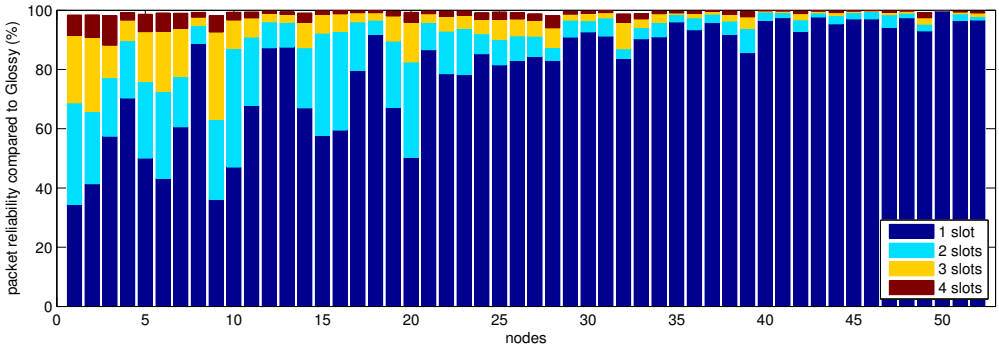


Figure 5.23: The expected packet reliability of Ripple compared to Glossy under different relay window size. Arena testbed, tx power = 0 dBm.

I collect hop distances to the initiator for each node and notice that the hop distance of a node is time variable on the same channel. Also, the distributions of the hop distances of the same node are different on different channels. To measure the expected degree of penalty on packet reliability under the case that I have the best static estimate of hop distance, I compute the percentage of packets received if I limit the relay window (reception chances) to a number of consecutive slots. Fig. 5.23 shows that when the relay window is equal to 2 slots (corresponding to transmission interval $i = 1$), the reliability of Ripple compared to Glossy is over 80% for most nodes, and in the worst case is still about 60%. Furthermore, if the relay window is increased to 3 slots ($i = 2$), the worst-case reliability of Ripple compared to Glossy goes up to 90%. My results further indicate that the reduction on reliability is smaller for smaller packet sizes. This shows that, with the increase of transmission interval, the reliability of Ripple improves quickly and approximates that of Glossy. This is Ripple's inherent trade-off between throughput and reliability.

5.3.6 *Testbed Evaluation*

I perform my evaluation on three testbeds of different size and topology: the Arena and Piloty testbeds at TU Darmstadt [GBKVL12], and the Flocklab at ETH Zurich [LFZ+13b]. The three testbeds have 57, 41 and 31 nodes, respectively. The main evaluation results are:

1. Two channels are sufficient for good performance of Ripple.
2. The baseline Ripple (without error coding) offers 2 to 3 times better throughput and energy efficiency, compared to Glossy.
3. Ripple with RS code achieves nearly 100% packet reliability, outperforming Glossy. The throughput is two to three times as large as that of the state-of-the-art data dissemination protocol Splash.
4. The measured throughput of Ripple is 80% to 90% of the analytical results.

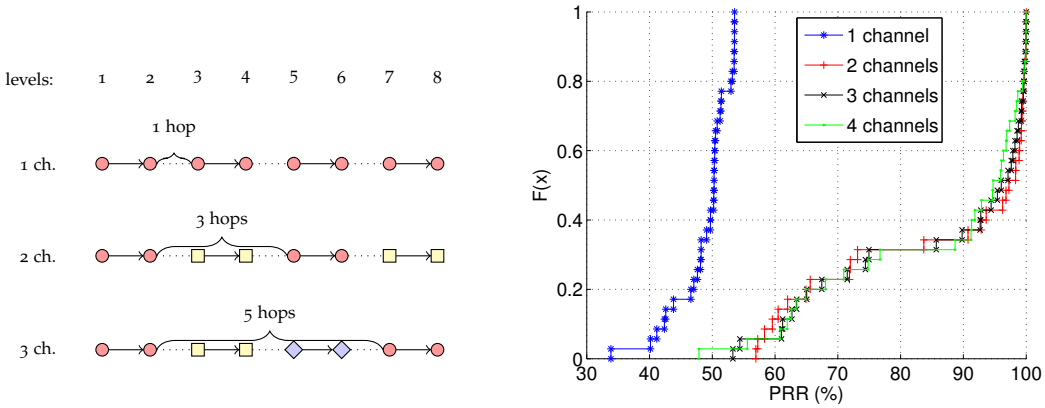
The memory footprints of Glossy, Ripple and Ripple with RS code are listed in Tab. 5.5. The extra RAM overhead of the baseline Ripple is moderate. Ripple with RS

code incurs more RAM overhead due to the RS code, still it fits in the 10K RAM of TelosB mote.

Table 5.5: Memory footprint comparison of Glossy, Ripple and Ripple-RS.

Program	Glossy	Ripple ($k = 20$)	Ripple-RS ($k = 10, n = 20$)
Code size (bytes)	16186	19838	23710
RAM (bytes)	449	546	1839

5.3.6.1 *The Impact of Channel Number on Ripple*



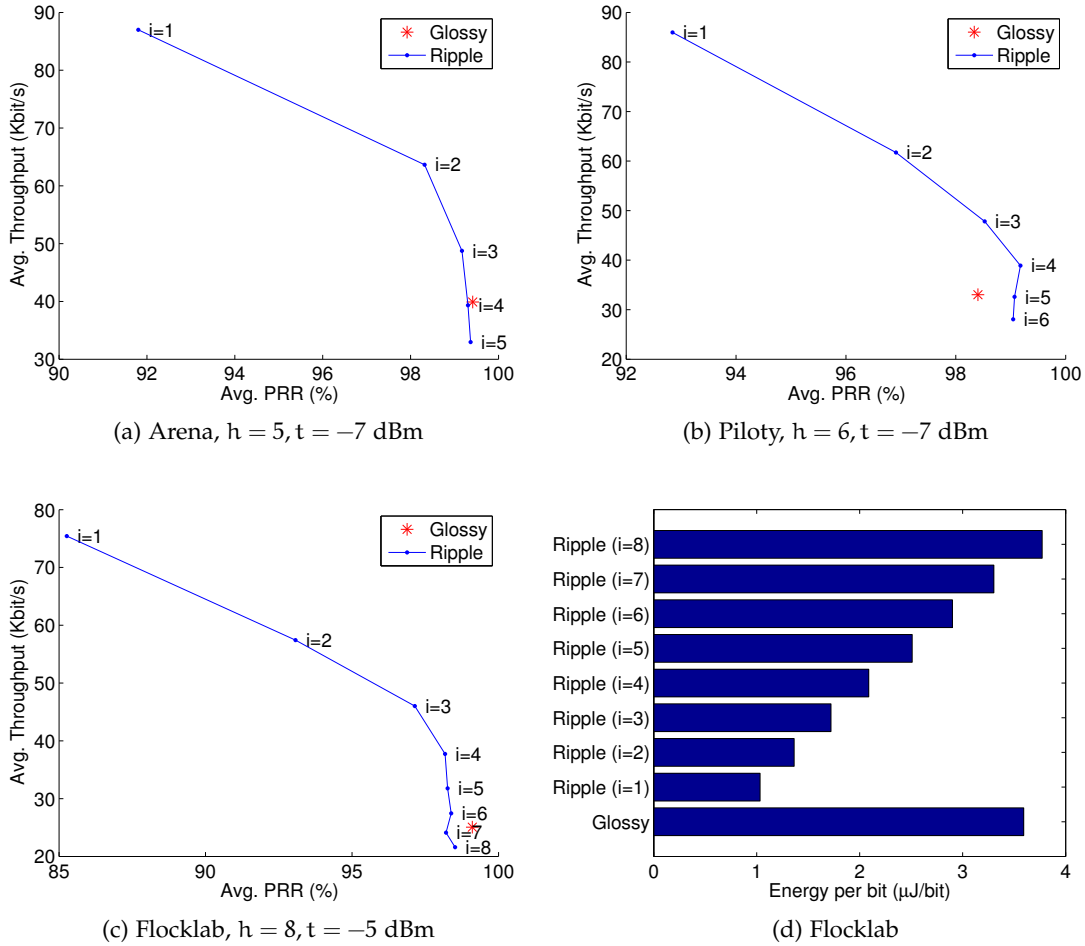
(a) Pipeline transmissions under different number of channels

(b) The CDF of PRR of all nodes for different number of channels

Figure 5.24: The impact of channel number on the reliability of Ripple. Transmission interval $i = 1$. In (a), level 1 is the initiator, level x are those nodes that are $(x-1)$ hops away from the initiator. Different shapes in (a) stand for different channels.

Ripple uses multiple channels to lower the mutual interference among simultaneous transmissions of different packets. The more channels I use, the less the mutual interference should be. This is reflected in Fig. 5.24(a): with transmission interval $i = 1$ and a single channel, the strongest interference to a receiver comes from the active transmitters located one hop away. Given y channels, the strongest interference comes from $y(i+1)-1$ hops away, which decreases very fast with the number of channels and transmission interval due to path loss.

I want to find out what is a practically sufficient number of channels. I run Ripple on the Piloty testbed (with the initiator at a corner, tx power $t = -7$ dBm and $i = 1$). Fig. 5.24(b) shows that with one channel, the interference is very strong, causing all nodes to have PRR below 55%. However, with two channels, PRR significantly improves, and is such that over 60% of the nodes have PRR over 90%. Further increase of the number of channels does not improve PRR. However, a potential advantage of a large number of channels is that the performance will degrade more smoothly when one channel is jammed or strongly interfered. Anyhow, for a good performance of Ripple, two channels are sufficient.



(a) Arena, $h = 5, t = -7$ dBm

(b) Piloty, $h = 6, t = -7$ dBm

(c) Flocklab, $h = 8, t = -5$ dBm

(d) Flocklab

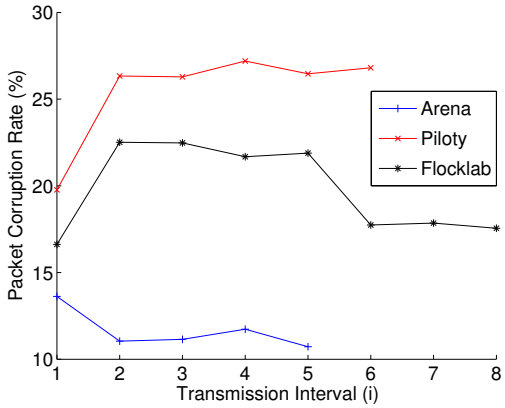
Figure 5.25: Performance of Ripple vs. Glossy for different transmission intervals (i).

5.3.6.2 *Performance of Ripple without Error Coding*

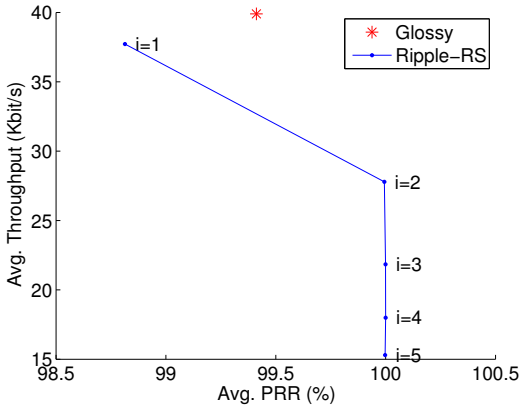
I compare the performance between Ripple and Glossy on three testbeds when there is no error coding. I use a middle transmission power t in order to get a network of relatively large diameter. The testbeds have different diameters h (listed in Fig. 5.25), which is defined as the distance to the initiator such that each node in the network has a distance less than or equal to h for over 95% of the cases when running Glossy (with the parameter *maximum transmission times* equal 2). h is used to determine the duration of a Glossy round and a Ripple round. The batch size is $k = 20$. The initiator node is at a corner of the network. The IEEE 802.15.4 channel #26 is used by Glossy, and two channels #26 and #25 are used by Ripple. Each run of Glossy or Ripple takes 30 minutes. I leverage the novel hardware-based high resolution power sampling provided by Flocklab [LFZ+13b] to get an accurate comparison of energy efficiency between the two protocols. I choose a sampling rate of 14,400 Hz.

The evaluation results are depicted in Fig. 5.25. In general, Ripple significantly increases the throughput compared to Glossy. [16] When the transmission interval

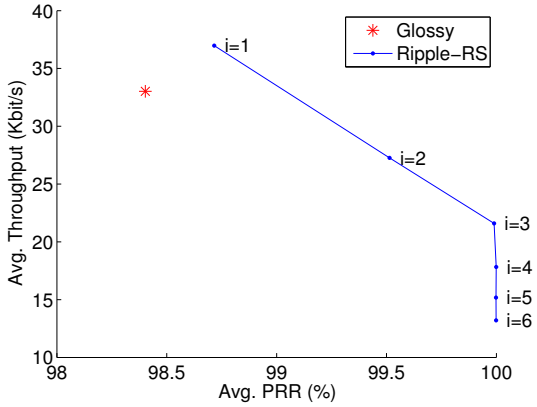
16 In the computation of throughput of Ripple, I have considered the synchronization overhead, which takes less than 1% of the time.



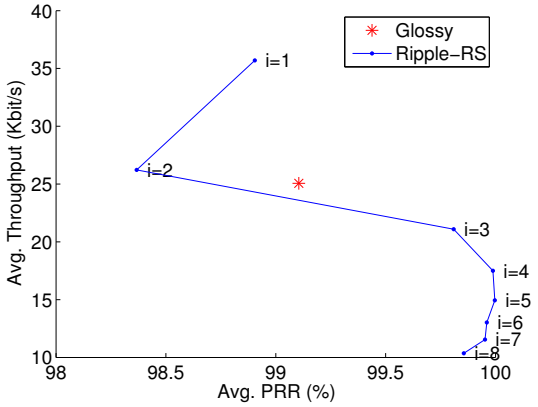
(a) Packet Corruption Rate of Ripple



(b) Arena, $h = 5, t = -7$ dBm



(c) Piloty, $h = 6, t = -7$ dBm



(d) Flocklab, $h = 8, t = -5$ dBm

Figure 5.26: Performance of Ripple with RS code vs. Glossy for different transmission intervals (i). Note, the figures (b)-(d) are of different scales to Fig. 5.25(a)-(c).

is $i = 1$, the improvements on the three testbeds are between 2.2x to 3.0x. The throughput improvement increases with the network diameter and decreases with i (recall that Ripple sends one packet every $i + 1$ slots). Moreover, the end-to-end reliability generally increases when i gets bigger, and this increase is particularly fast for $i \leqslant 3$. With the further increase of i to $h - 1$, Ripple converges to Glossy. Therefore, a nice feature of Ripple is that by tuning the transmission interval, I can trade between high throughput and high reliability, which can suit a large spectrum of network broadcast applications with different QoS requirements.

Fig. 5.25(d) compares the energy efficiency of Glossy and Ripple. The metric chosen is the energy per bit, i.e., the total energy consumed by the whole network over the total number of bits of the distinct packets received by all nodes. Ripple is over 3 times more energy efficient than Glossy when $i = 1$. With the further increase of i, it uses more energy, but it is still more energy efficient than Glossy even at $i = h - 1$ where the reliability and throughput of the two protocols are similar. The higher energy efficiency of Ripple is expected due to the higher throughput in the same radio-on time.

5.3.6.3 *Performance of Ripple with RS Erasure Code*

If the reliability requirement is higher than what can be offered by the baseline Ripple, I can apply error coding. Protocols based on concurrent transmission such as Ripple display a large amount of corrupted packets, which are mainly due to the imperfect constructive interference. For the three testbeds, the packet corruption rate (corrupted packets over all packets passing preamble and SFD checks) ranges from 10% to 30% (Fig.5.26(a)). I use the RS erasure code to compensate for that. In the evaluation, I encode 10 source packets (k) into 20 packets (n) in each batch. The error coding improves the reliability, but the trade-off is that it decreases the throughput due to the coding redundancy ($n > k$), and the extra time for encoding and decoding. Except for the application of the RS code, the evaluation setting is the same as before.

Through experiments, I found that the worst-case encoding and decoding times are both 52 milliseconds, when $k = 10, n = 20$ and packet size is at maximum. Therefore, in each batch, I assign an extra 52 ms for the encoding and decoding of the RS code. Considering that transmitting a packet of maximum length takes about 5 ms, the encoding/decoding of such a packet takes 5.2 ms, roughly the same time as the communication.

The performance of Ripple with RS code is shown in Fig. 5.26(b)-(d). Its predominant feature is the extremely high end-to-end reliability. For all three testbeds, the PRRs are better than those of Glossy and very near to 100% at big transmission intervals. Ripple with RS code may attain the sweet spot of higher reliability and larger throughput than Glossy (Fig. 5.26(c)). Compared to the state-of-the-art data dissemination protocol Splash, which achieved a throughput of 10 Kbits/s as reported in [DCL13b], Ripple obtains two to three times higher throughput when the transmission interval $i \leqslant 2$, at the slight cost of 1% reduction in reliability. The throughput also transitively outperforms that of the popular data dissemination protocol Deluge T2, which is outperformed by Splash for over 10 times, as reported in [DCL13b]. Another advantage of Ripple in comparison with the other data dissemination protocols is that Ripple has bounded end-to-end latency due to the time bounded data rounds. This makes Ripple suitable to latency-sensitive applications, such as industrial automation.

However, the decrease in throughput is quite big compared to the baseline Ripple. One reason is the long encoding/decoding time. The throughput can be improved by using sensor nodes with faster CPUs (the Ripple implementation on TelosB uses a CPU clock of 4 MHz) or by selecting a higher code rate matching the end-to-end communication quality.

5.3.6.4 *Comparison of Evaluation and Theory*

The theoretical throughput derived in Sec. 5.3.4 is the upper bound that Ripple can achieve. Fig. 5.27 depicts how close the evaluation results can come to the theory. The measured throughput achieves 80% to 90% of the theoretical throughput on all three testbeds. The 10% to 20% decrease is due to multiple factors: 1) the overhead of the sync rounds, 2) the guard interval at the beginning of the data rounds, and 3) most importantly, the implementation differing from the analytical assumption in that the data rounds on all nodes are synchronized. This last factor leads to easier implementation but causes some constant time overhead in each data round. In future work, I will investigate displaced data rounds on different network levels. My results

exhibit a distinct trend: the measured throughput approaches the theoretical value when either the transmission interval goes up or when the RS code is applied. The underlying reason is the same in both cases — when the duration of a data round increases, the constant overhead of the synchronized data rounds becomes negligible. Furthermore, as expected by the theory (c.f. Sec. 5.3.4), the throughput of Ripple is independent of the network size h.

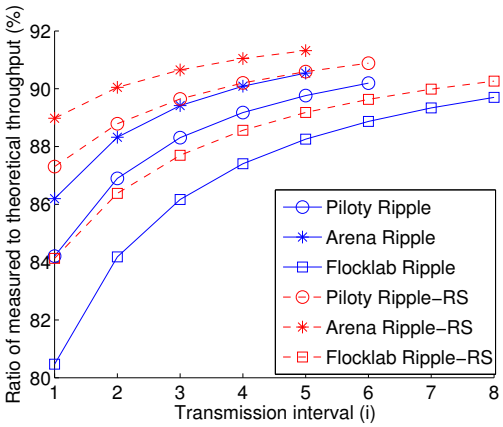


Figure 5.27: The ratio of the measured average throughput to the theoretical average throughput as derived in Sec. 5.3.4. The parameters used in the computation of the theoretical throughput are: the packet size $l = 127$ bytes, the slot time $t_{slot} = 4.62$ ms, the encoding/decoding time $t_{ed} = 52$ ms and the reliability R is the measured PRR.

5.3.7 *Conclusion*

Ripple is a high-throughput, reliable and energy efficient network flooding protocol for WSNs. It is based on the physical layer feature of constructive interference, and adds pipeline transmission on multiple channels and error coding to the state-of-the-art. The pipeline transmission raises the throughput over 80 Kbits/s in the testbed evaluation, a threefold increase compared to Glossy. It comes close to the theoretical upper bound of half-duplex radios. Ripple also increases the energy efficiency by a factor of three, compared to Glossy. By tuning the transmission interval, Ripple balances between high reliability and high throughput, suiting a large spectrum of QoS requirements. To make Ripple suitable for reliable data dissemination, I can apply Reed-Solomon erasure code to it. This pushes the reliability over that of Glossy, very near to 100%, but at the cost of reduced throughput over plain Ripple. Still, the throughput doubles, or even triples that of the state-of-the-art data dissemination protocol Splash. Finally, contrary to Glossy, the throughput of Ripple is shown by analysis and evaluation to be independent of network size.

5.4 SUMMARY

Concurrent transmission is a physical layer technique with high potential in improving quality of service in wireless communication. Its strength is drawn mainly from the constructive interference (constructive addition of the signals from multiple transmit-

ters) and the receive diversity, where multiple receivers are allowed to hear from the same set of transmitters at the same time. Concurrent transmission is helpful to wireless communication because the traditional one-to-one communication is unreliable. This is particularly true for wireless sensor networks. Therefore, the key to reliable low latency communication is to abandon the Single Transmitter Single Receiver (STSR) communication paradigm and to adopt the transmit/receive diversity. Suppose m transmitters and n receivers are active at the same time, there are effectively $m \cdot n$ STSR links active simultaneously. The large amount of links can adequately compensate the high loss rate of each link.

This chapter starts with the modeling of concurrent transmission which deepens our understanding of it and provides a useful tool for the prediction of packet reception. After that it demonstrates that concurrent transmission can be successfully applied to multi-rate periodic monitoring and control systems. The proposed Sparkle protocol can adaptively control the QoS of end-to-end communication flows by controlling the topology with capture effect and by controlling the transmission power. Finally I propose the Ripple protocol, which significantly improves the QoS of one-to-all broadcast by incorporating pipeline concurrent transmission on multiple channels and error coding. A prominent advantage of concurrent transmission is that it is routingless and is therefore more robust, reliable and resilient while incurring no routing overhead. The technique has recently appeared and is gradually maturing.

6 CONCLUSIONS AND OUTLOOK

6.1 CONCLUSIONS

The main focus of this thesis is on improving QoS in communication for industrial wireless sensor networks. The deficiency in QoS in communication is the biggest obstacle to the goal of wide adoption of WSN in industrial automation. In this thesis, I have proposed solutions that advance the state-of-the-art, and thus bring us a step further to the goal.

The thesis enhances QoS in communication by performing research in two areas. The first area is the centralized scheduling for WirelessHART protocol and the second area is using concurrent transmission for routing-free reliable, low-latency and energy efficient communication.

Regarding centralized scheduling, I first investigate the problem of WirelessHART-based convergecast/distribution scheduling. I propose a novel busy-sender-first heuristic that is based on the simple idea of prioritizing the busy sender, i.e., the node with the largest number of remaining transmissions. It is demonstrated by extensive simulation to be significantly better than the state-of-the-art heuristic in both schedule length and memory consumption. Second, I investigate a very general scheduling problem of periodic mesh-based multi-flow communication with hard deadlines. I propose a lightweight and effective scheduling heuristic Least Laxity First (LLF), which offers the highest schedulability rate among all heuristics evaluated, very low execution time and memory overhead. Furthermore, I propose the opportunistic aggregation scheme which works seamlessly with any heuristic and substantially increase the schedulability. I also propose the repetitive scheduling scheme that is also compatible with any heuristic, and leads to low and scalable cost of schedule table and execution time. It works ideally when the periods of all flows are harmonic and their deadlines are implicit (equal to the respective periods).

The recently appeared transmit/receive diversity technique of concurrent transmission in WSN has shown great potentials in improving QoS in communication of WSN. The thesis is the first effort in applying concurrent transmission to industrial WSN. For better understanding of the various physical layer phenomena behind concurrent transmission, I propose an accurate prediction model for packet reception. Then I propose Sparkle, a WSN control network based on concurrent transmission, for periodic multi-loop control systems. It controls the communication QoS of each flow by combining an efficient topology control based on capture effect with the transmission power control. Finally, I propose Ripple, which extends the Glossy network flooding with pipelined concurrent transmission on multiple channels and forward error correction. It significantly improves on the throughput, energy efficiency and reliability, compared to the state-of-the-art protocol Splash and Glossy. All the works on concurrent transmission are verified by evaluations on public available real-world testbeds.

Generally, both parts of the thesis have demonstrated that IEEE 802.15.4 based WSN, despite its limited capability in communication, can be effectively applied to industrial automation with strict requirements on communication QoS, through high quality scheduling heuristics and the physical layer diversity technique of concurrent transmission.

6.2 OUTLOOK

The thesis contributes in facilitating the application of WSN to industrial automation. However, the problem has not yet been fully solved. This problem and the more general problem of how to improve QoS in wireless communication will have a fundamental impact on the future of communication. In the following, I enumerate a number of interesting and important open problems that are to be investigated:

1. How do different wireless standards compare with each other? I am not aware of a complete comparison of different wireless standards for industrial applications in terms of different QoS metrics, such as latency, throughput, reliability and energy efficiency. Quite often, the researchers presume a standard, such as IEEE 802.15.4 or IEEE 802.11. Such choices are somehow arbitrary. A complete comparison of wireless standards can substantiate or disprove a choice.

2. How can we support priority arbitration in wireless communication? The widely used Controller Area Network (CAN) bus for automotive and industrial automation applications has a unique feature of message priority arbitration. It works as follows: the transmissions of different messages are synchronized on the start bit. The arbitration is performed on the message identifier following the start bit, where a bit zero dominates a bit one. For each station, the transmission and listen happen at the same time. If a node hears a different bit from the corresponding one in its message identifier, it loses the right to continue to send [LMT05]. Therefore, with priority arbitration, the CAN bus can give guarantees on message priority.

   The feature of priority arbitration are important for industrial applications. For example, some control loops may be more important than the others for the correct operation of a system. The priority arbitration is much more difficult to realize in the wireless communication than in the wired communication because the channel activity perceived by different nodes are different. Therefore, a perfect solution such as that of CAN bus may be unavailable. Some solutions for wireless networks do exist. The MAC layer solution of [PKB03] assigns different waiting time to nodes of different priorities after the channel is detected as free. This solution may incur long time overhead. The physical layer solution by Huang et. al [HYX13] is more efficient, but it needs two radios for simultaneous transmission and listen. How to design an effective priority arbitration mechanism in wireless communication is worth investigation.

3. How can we support both event-based and periodic traffic? In industrial monitoring and control systems, the periodic traffic such as the transmission of sensor data may co-exist with the event-based traffic such as alerts. Periodic traffic can normally be pre-scheduled since it is static or semi-static while the

event-based traffic is unpredictable. How to support these two types of traffic in the same system is an open problem. It is related to the last problem of priority arbitration since the event-based traffic normally has higher priority than the periodic traffic.

4. How to utilize multi-radio technology to improve latency and throughput in wireless communication? The predominant sensor nodes available today are equipped with only one radio. Therefore, the minimum communication latency of a packet through n hops will be n times of the transmission duration of the packet. The maximum throughput over multi-hop will be half of the data rate of the radio. However, as illustrated in [KAHH06], a multi-radio WSN can reduce the latency to roughly n times of the transmission duration of a packet header, by using multiple channels. I argue that the throughput can be increased for n fold in such a setting and if the network is highly time synchronized, the latency can be even decreased to n-bit time. Thus the latency and throughput can be both heavily improved, which makes the industrial wireless networks more competitive to the wired counterpart by approaching or even surpassing its performance.

In conclusion, the optimization and guarantee of QoS metrics are key to the adoption of wireless technology in industrial automation. The thesis shows that real-time scheduling and concurrent transmission provide two enabling technologies for this purpose. The evaluation on real-world testbeds confirms their excellent performance and foresees a bright future for the full penetration of wireless technology in industrial automation.

BIBLIOGRAPHY

[ÅGB11] J. Åkerberg, M. Gidlund, and M. Bjorkman: *Future Research Challenges in Wireless Sensor and Actuator Networks Targeting Industrial Automation*. In *Proceedings of the 9th IEEE International Conference on Industrial Informatics (INDIN)*, 2011.

[ÅM08] K. Åström and R. Murray: *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008.

[ASSC02] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci: *Wireless Sensor Networks: a Survey*. Computer Networks, 38(4):393–422, 2002.

[BB05] E. Bini and G. C. Buttazzo: *Measuring the Performance of Schedulability Tests*. Real-Time Systems, 30(1-2):129–154, 2005.

[BH80] T. Beyer and S. M. Hedetniemi: *Constant Time Generation of Rooted Trees*. SIAM Journal on Computing, 9(4):706–712, 1980.

[CC1107] *Chipcon CC1000 Datasheet*. Revision A edition, 2007.

[CC08] S. Chalasani and J. M. Conrad: *A Survey of Energy Harvesting Sources for Embedded Systems*. In *Proceedings of the IEEE Southeastcon 2008*, 2008.

[CC213] *Chipcon CC2420 Datasheet*. Revision C edition, 2013.

[CCD+11] M. Ceriotti, M. Corrà, L. D'Orazio, R. Doriguzzi, D. Facchin, S. Guna, G. P. Jesi, R. L. Cigno, L. Mottola, A. L. Murphy, M. Pescalli, G. P. Picco, D. Pregnolato, and C. Torghele: *Is There Light at the Ends of the Tunnel? Wireless Sensor Networks for Adaptive Lighting in Road Tunnels*. In *Proceedings of the 10th International Conference on Information Processing in Sensor Networks (IPSN)*, 2011.

[CCT+13] D. Carlson, M. Chang, A. Terzis, Y. Chen, and O. Gnawali: *Forwarder Selection in Multi-transmitter Networks*. In *Proceedings of the 9th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2013.

[CG06] L. Cucu and J. Goossens: *Feasibility Intervals for Fixed-Priority Real-Time Scheduling on Uniform Multiprocessors*. In *Proceedings of 11th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2006.

[CG07] L. Cucu and J. Goossens: *Feasibility Intervals for Multiprocessor Fixed-priority Scheduling of Arbitrary Deadline Periodic Systems*. In *Proceedings of the Design, Automation and Test in Europe Conference and Exposition (DATE)*, 2007.

[CLR07] O. Chipara, C. Lu, and G.-C. Roman: *Real-Time Query Scheduling for Wireless Sensor Networks*. In *Proceedings of the 28th IEEE Real-Time Systems Symposium (RTSS)*, 2007.

[CNM10] D. Chen, M. Nixon, and A. Mok: *WirelessHART Real-time Mesh Network for Industrial Automation*. Springer, 2010.

[CT11] Y. Chen and A. Terzis: *On the Implications of the Log-normal Path Loss Model: an Efficient Method to Deploy and Move Sensor Motes*. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2011.

[CZSB02] M. Caccamo, L. Y. Zhang, L. Sha, and G. C. Buttazzo: *An Implicit Prioritized Access Protocol for Wireless Sensor Networks*. In *Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS)*, 2002.

[DB11a] R. I. Davis and A. Burns: *A Survey of Hard Real-time Scheduling for Multiprocessor Systems*. ACM Computing Surveys, 43(4):35, 2011.

[DB11b] R. I. Davis and A. Burns: *Improved Priority Assignment for Global Fixed Priority Pre-emptive Scheduling in Multiprocessor Real-time Systems*. Real-Time Systems, 47(1):1–40, 2011.

[DC14] M. Doddavenkatappa and M. C. Chan: *P^3: a Practical Packet Pipeline using Synchronous Transmissions for Wireless Sensor Networks*. In *Proceedings of the 13th International Conference on Information Processing in Sensor Networks (IPSN)*, 2014.

[DCL13a] M. Doddavenkatappa, M. C. Chan, and B. Leong: *Splash: Fast Data Dissemination with Constructive Interference in Wireless Sensor Networks*. In *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2013.

[DCL13b] M. Doddavenkatappa, M. C. Chan, and B. Leong: *Splash: Fast Data Dissemination with Constructive Interference in Wireless Sensor Networks*. In *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2013.

[DGV04] A. Dunkels, B. Grönvall, and T. Voigt: *Contiki – A Lightweight and Flexible Operating System for Tiny Networked Sensors*. In *Proceedings of the 29th IEEE Conference on Local Computer Networks (LCN)*, 2004.

[Die05] R. Diestel: *Graph Theory*. Springer, third edition, 2005.

[DLT+12] S. Dawson-Haggerty, S. Lanzisera, J. Taneja, R. Brown, and D. E. Culler: *@scale: Insights from a Large, Long-lived Appliance Energy WSN*. In *Proceedings of the 11th International Conference on Information Processing in Sensor Networks (IPSN)*, 2012.

[DMEST08] P. Dutta, R. Musăloiu-E, I. Stoica, and A. Terzis: *Wireless ACK Collisions Not Considered Harmful*. In *Proceedings of the 7th ACM Workshop on Hot Topics in Networks (HotNets)*, 2008.

[DÖTH07] A. Dunkels, F. Österlind, N. Tsiftes, and Z. He: *Software-based On-line Energy Estimation for Sensor Nodes*. In *Proceedings of the 4th Workshop on Embedded Networked Sensors (EmNets)*, 2007.

[EV10] S. C. Ergen and P. Varaiya: *TDMA Scheduling Algorithms for Wireless Sensor Networks*. Wireless Networks, 16(4):985–997, 2010.

[FPW98] G. F. Franklin, J. D. Powell, and M. L. Workman: *Digital Control of Dynamic Systems*. Addison Wesley Longman, Inc., third edition, 1998.

[FZMT12] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele: *Low-power Wireless Bus*. In *Proceedings of the 10th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2012.

[FZTS11] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh: *Efficient Network Flooding and Time Synchronization with Glossy*. In *Proceedings of the 10th International Conference on Information Processing in Sensor Networks (IPSN)*, 2011.

[GBKVL12] P. E. Guerrero, A. P. Buchmann, A. Khelil, and K. Van Laerhoven: *TUDμNet, a Metropolitan-Scale Federation of Wireless Sensor Network Testbeds*. In *Proceedings of the 9th European Conference on Wireless Sensor Networks (EWSN)*, 2012.

[GFJ+09] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis: *Collection Tree Protocol*. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2009.

[GZH08] S. Gandham, Y. Zhang, and Q. Huang: *Distributed Time-optimal Scheduling for Convergecast in Wireless Sensor Networks*. Computer Networks, 52(3):610–629, 2008.

[HAR07] *WirelessHART Specifications*. http://www.hartcomm2.org, 2007.

[HC04] J. W. Hui and D. E. Culler: *The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale*. In *Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[HCF07] *HART Field Communication Protocol Specification, Revision 7.0*. HART Communication Foundation, 2007.

[Hor74] W. A. Horn: *Some Simple Scheduling Algorithms*. Naval Research Logistics Quarterly, 21(1):177–185, 1974.

[HYX13] P. Huang, X. Yang, and L. Xiao: *WiFi-BA: Choosing Arbitration over Backoff in High Speed Multicarrier Wireless Networks*. In *Proceedings of the 32nd IEEE International Conference on Computer Communications (INFOCOM)*, 2013.

[IGK11] O. D. Incel, A. Ghosh, and B. Krishnamachari: *Scheduling Algorithms for Tree-Based Data Collection in Wireless Sensor Networks*. In *Theoretical Aspects of Distributed Computing in Sensor Networks*. Springer, 2011.

[ISA08] *The ISA100 Standards, Overview and Status*. http://www.isa.org, 2008.

[ISA09] *Wireless Systems for Industrial Automation: Process Control and Related Applications*. ISA-100.11a-2009 Standard, 2009.

[KAHH06] M. Kohvakka, T. Arpinen, M. Hännikäinen, and T. D. Hämäläinen: *High-performance Multi-radio WSN Platform*. In *Proceedings of the 2nd International Workshop on Multi-hop Ad Hoc Networks: From Theory to Reality (REALMAN)*, 2006.

[KFD+07] S. Kim, R. Fonseca, P. Dutta, A. Tavakoli, D. E. Culler, P. Levis, S. Shenker, and I. Stoica: *Flush: a Reliable Bulk Transport Protocol for Multihop Wireless Networks*. In *Proceedings of the 5th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2007.

[KPC+07] S. Kim, S. Pakzad, D. E. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon: *Health Monitoring of Civil Infrastructures using Wireless Sensor Networks*. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN)*, 2007.

[KW59] J. E. Kelley and M. R. Walker: *Critical-path Planning and Scheduling*. In *Proceedings of Eastern Joint IRE-AIEE-ACM Computer Conference*, 1959.

[Lee94] S. K. Lee: *On-line Multiprocessor Scheduling Algorithms for Real-time Tasks*. In *Proceedings of the IEEE Region 10's 9th Annual International Conference (TENCON)*, volume 2, 1994.

[LFZ13a] O. Landsiedel, F. Ferrari, and M. Zimmerling: *Chaos: a Versatile and Efficient Communication Primitive for All-to-all Data Sharing and In-network Processing at Scale*. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2013.

[LFZ+13b] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel: *FlockLab: a Testbed for Distributed, Synchronized Tracing and Profiling of Wireless Embedded Systems*. In *Proceedings of the 12th International Conference on Information Processing in Sensor Networks (IPSN)*, 2013.

[Liu00] J. W. S. W. Liu: *Real-Time Systems*. Prentice Hall PTR, 1st edition, 2000.

[LLL+09] C. M. Liang, J. Liu, L. Luo, A. Terzis, and F. Zhao: *RACNet: a High-fidelity Data Center Sensing Network*. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2009.

[LLWC03] P. Levis, N. Lee, M. Welsh, and D. E. Culler: *TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications*. In *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[LMT05] F. Lian, J. R. Moyne, and D. M. Tilbury: *Network Protocols for Networked Control Systems*. In *Handbook of Networked and Embedded Control Systems*, pages 651–676. 2005.

[LMW+16] B. Li, Y. Ma, T. Westenbroek, C. Wu, H. Gonzalez, and C. Lu: *Wireless Routing and Control: a Cyber-Physical Case Study*. In *Proceedings of the ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2016.

[LPCS04] P. Levis, N. Patel, D. E. Culler, and S. Shenker: *Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks*. In *Proceedings of the 1st USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2004.

[LW09] J. Lu and K. Whitehouse: *Flash Flooding: Exploiting the Capture Effect for Rapid Flooding in Wireless Sensor Networks*. In *Proceedings of the 28th IEEE International Conference on Computer Communications (INFOCOM)*, 2009.

[LZ06] I. D. Landau and G. Zito: *Digital Control Systems: Design, Identification and Implementation*. Springer, 2006.

[MET08] R. Musaloiu-E. and A. Terzis: *Minimising the Effect of WiFi Interference in 802.15.4 Wireless Sensor Networks*. International Journal of Sensor Networks (IJSNet), 3(1):43–54, 2008.

[MJD08a] R. Maheshwari, S. Jain, and S. R. Das: *A Measurement Study of Interference Modeling and Scheduling in Low-power Wireless Networks*. In *Proceedings of the 6th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2008.

[MJD08b] R. Maheshwari, S. Jain, and S. R. Das: *On Estimating Joint Interference for Concurrent Packet Transmissions in Low Power Wireless Networks*. In *Proceedings of the 3rd ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH)*, 2008.

[MOS] MOSEK ApS: *The MOSEK optimization software (version 6.0, revision 114)*. `http://www.mosek.com/`.

[Mot04] Moteiv Corporation: *Telos-B Datasheet*, 2004.

[MPFJ10] P. D. Marco, P. G. Park, C. Fischione, and K. H. Johansson: *TREnD: A Timely, Reliable, Energy-Efficient and Dynamic WSN Protocol for Control Applications*. In *Proceedings of the IEEE International Conference on Communications (ICC)*, 2010.

[MT07] J. R. Moyne and D. Tilbury: *The Emergence of Industrial Control Networks for Manufacturing Control, Diagnostics, and Safety Data*. Proceedings of the IEEE, 95(1):29–47, 2007.

[NRR12] M. Nixon and T. Round Rock: *A Comparison of WirelessHART and ISA100.11a*. white paper, July, 2012.

[OBB+13] T. O'Donovan, J. Brown, F. Büsching, A. Cardoso, J. Cecílio, J. M. do Ó, P. Furtado, P. Gil, A. Jugel, W. Pöttner, U. Roedig, J. S. Silva, R. Silva, C. J. Sreenan, V. Vassiliou, T. Voigt, L. C. Wolf, and Z. Zinonos: *The GINSENG System for Wireless Monitoring and Control: Design and Deployment Experiences*. ACM Transactions on Sensor Networks (TOSN), 10(1):4, 2013.

[PC11] S. Petersen and S. Carlsen: *WirelessHART versus ISA100.11a: The Format War Hits the Factory Floor*. IEEE Industrial Electronics Magazine, 5(4):23–34, 2011.

[PG07] J. Paek and R. Govindan: *RCRT: Rate-controlled Reliable Transport for Wireless Sensor Networks*. In *Proceedings of the 5th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2007.

[PJJ+14] J. Park, J. Jeong, H. Jeong, C.-J. M. Liang, and J. Ko: *Improving the Packet Delivery Performance for Concurrent Packet Transmissions in WSNs*. IEEE Communications Letters, 18(1):58–61, 2014.

[PKB03] W. Pattara-Atikom, P. Krishnamurthy, and S. Banerjee: *Distributed Mechanisms for Quality of Service in Wireless LANs*. IEEE Wireless Communications, 10(3):26–34, 2003.

[PN09] C. E. Pereira and P. Neumann: *Industrial Communication Protocols*. In S. Y. Nof (editor): *Springer Handbook of Automation*, chapter 56. Springer, second edition, 2009.

[PSC05] J. Polastre, R. Szewczyk, and D. E. Culler: *Telos: Enabling Ultra-low Power Wireless Research*. In *Proceedings of the 4th International Conference on Information Processing in Sensor Networks (IPSN)*, 2005.

[Rap01] T. Rappaport: *Wireless Communications: Principles and Practice*. Prentice Hall PTR, 2nd edition, 2001.

[RCBG10] B. Raman, K. Chebrolu, S. Bijwe, and V. Gabale: *PIP: a Connection-oriented, Multi-hop, Multi-channel TDMA-based MAC for High Throughput Bulk Transfer*. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2010.

[Riz97] L. Rizzo: *Effective Erasure Codes for Reliable Computer Communication Protocols*. Computer Communication Review, 27(2):24–36, 1997.

[RM89] T. Rappaport and C. McGillem: *UHF Fading in Factories*. IEEE Journal on Selected Areas in Communications, 7(1):40–48, 1989.

[SBR10] P. Suriyachai, J. Brown, and U. Roedig: *Time-Critical Data Delivery in Wireless Sensor Networks*. In *Proceedings of the 6th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2010.

[SG11] R. Sugihara and R. K. Gupta: *Clock Synchronization with Deterministic Accuracy Guarantee*. In *Proceedings of the 8th European Conference on Wireless Sensor Networks (EWSN)*, 2011.

[SKH06] D. Son, B. Krishnamachari, and J. S. Heidemann: *Experimental Study of Concurrent Transmission in Wireless Sensor Networks*. In *Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2006.

[Soc06] I. C. Society: *IEEE Standard 802.15.4*. 2006 edition, 2006.

[SS11] M. Santina and A. R. Stubberud: *Sample-Rate Selection*. In W. S. Levine (editor): *The Control Handbook – Control System Fundamentals*, chapter 15. CRC Press, second edition, 2011.

[SW07] H. Strese and B. Wybranski: *From Mainframe Computers to Microsystems-Factory Automation for 30 Years*. MST NEWS, 2:6, 2007.

[SXLC10] A. Saifullah, Y. Xu, C. Lu, and Y. Chen: *Real-Time Scheduling for WirelessHART Networks*. In *Proceedings of the 31st IEEE Real-Time Systems Symposium (RTSS)*, 2010.

[SXLC11a] A. Saifullah, Y. Xu, C. Lu, and Y. Chen: *End-to-End Delay Analysis for Fixed Priority Scheduling in WirelessHART Networks*. In *Proceedings of the 17th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2011.

[SXLC11b] A. Saifullah, Y. Xu, C. Lu, and Y. Chen: *Priority Assignment for Real-time Flows in WirelessHART Networks*. In *Proceedings of the 23rd Euromicro Conference on Real-Time Systems (ECRTS)*, 2011.

[SZJ09] P. Soldati, H. Zhang, and M. Johansson: *Deadline-constrained Transmission Scheduling and Data Evacuation in WirelessHART Networks*. In *Proceedings of European Control Conference (ECC)*, 2009.

[Tex11] Texas Instruments: *MSP430F15x, MSP430F16x, MSP430F161x Mixed Signal Microcontroller*. Revision G edition, 2011.

[Tho05] J.-P. Thomesse: *Fieldbus Technology in Industrial Automation*. Proceedings of the IEEE, 93(6):1073 –1101, 2005.

[TJV+08] E. Tanghe, W. Joseph, L. Verloock, L. Martens, H. Capoen, K. V. Herwegen, and W. Vantomme: *The Industrial Indoor Channel: Large-scale and Temporal Fading at 900, 2400, and 5200 MHz*. IEEE Transactions on Wireless Communications, 7(7):2740–2751, 2008.

[WHC+13] Y. Wang, Y. He, D. Cheng, Y. Liu, and X.-Y. Li: *TriggerCast: Enabling Wireless Consrucive Collisions*. In *Proceedings of the 32nd IEEE International Conference on Computer Communications (INFOCOM)*, 2013.

[WHM+12] Y. Wang, Y. He, X. Mao, Y. Liu, Z. Huang, and X.-Y. Li: *Exploiting Constructive Interference for Scalable Flooding in Wireless Networks*. In *Proceedings of the 31st IEEE International Conference on Computer Communications (INFOCOM)*, 2012.

[WWJ+05] K. Whitehouse, A. Woo, F. Jiang, J. Polastre, and D. Culler: *Exploiting the Capture Effect for Collision Detection and Recovery*. In *Proceedings of the 2nd Workshop on Embedded Networked Sensors (EmNets)*, 2005.

[YB11] R. Yedavalli and R. Belapurkar: *Application of Wireless Sensor Networks to Aircraft Control and Health Management Systems*. Journal of Control Theory and Applications, 9(1):28–33, 2011.

[YH12a] D. Yuan and M. Hollick: *Optimization and Scheduling of Wireless Sensor Networks for Periodic Control Systems*. In *Proceedings of the 11th GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze" (FGSN)*, 2012.

[YH12b] D. Yuan and M. Hollick: *Tree-based Multi-Channel Convergecast in Wireless Sensor Networks*. In *Proceedings of the 13th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2012.

[YH13] D. Yuan and M. Hollick: *Let's talk together: Understanding concurrent transmission in wireless sensor networks*. In *Proceedings of the 38th IEEE Conference on Local Computer Networks (LCN)*, 2013.

[YH15] D. Yuan and M. Hollick: *Ripple: High-throughput, reliable and energy-efficient network flooding in wireless sensor networks*. In *Proceedings of the 16th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2015.

[YRH14] D. Yuan, M. Riecker, and M. Hollick: *Making 'Glossy' Networks Sparkle: Exploiting Concurrent Transmissions for Energy Efficient, Reliable, Ultra-Low Latency Communication in Wireless Control Networks*. In *Proceedings of the 11th European Conference on Wireless Sensor Networks (EWSN)*, 2014.

[ZBP01] W. Zhang, M. S. Branicky, and S. M. Phillips: *Stability of Networked Control Systems*. IEEE Control Systems Magazine, 21:84–99, 2001.

[ZHZ+11] G. Zheng, D. Han, R. Zheng, C. Schmitz, and X. Yuan: *A Link Quality Inference Model for IEEE 802.15.4 Low-Rate WPANs*. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, 2011.

[ZK04] M. Zuniga and B. Krishnamachari: *Analyzing the Transitional Region in Low Power Wireless Links*. In *Proceedings of the 1st IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, 2004.

[ZÖS+10] H. Zhang, F. Österlind, P. Soldati, T. Voigt, and M. Johansson: *Rapid Convergecast on Commodity Hardware: Performance Limits and Optimal Policies*. In *Proceedings of the 7th IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, 2010.

[ZSJ09a] H. Zhang, P. Soldati, and M. Johansson: *Efficient Link Scheduling and Channel Hopping for Convergecast in WirelessHART Networks*. Technical Report TRITA-EE 2009:050, Royal Institute of Technology Automatic Control Lab, 2009.

[ZSJ09b] H. Zhang, P. Soldati, and M. Johansson: *Optimal Link Scheduling and Channel Assignment for Convergecast in Linear WirelessHART Networks*. In *Proceedings of the 7th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2009.

[ZSJ13] H. Zhang, P. Soldati, and M. Johansson: *Performance Bounds and Latency-optimal Scheduling for Convergecast in WirelessHART Networks*. IEEE Transactions on Wireless Communications, 12(6):2688–2696, 2013.

LIST OF ACRONYMS

6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
ACK	Acknowledgement
ASR	Action Slot Rate
BER	Bit Error Rate
BIP	Binary Integer Programming
CAN	Controller Area Network
CI	Constructive Interference
CLLF	Conflict-aware Least Laxity First
CRC	Cyclic Redundancy Check
CTP	Collection Tree Protocol
DCO	Digitally Controlled Oscillator
DM	Deadline Monotonic
DSSS	Direct Sequence Spread Spectrum
EDF	Earliest Deadline First
EDZL	Earliest Deadline Zero Laxity
EPD	Earliest Proportional Deadline
FSK	Frequency Shift Keying
GW	Gateway
HCF	HART Communication Foundation
IC	Integrated Circuits
IETF	Internet Engineering Task Force
IP	Integer Programming
IPv6	Internet Protocol version 6
ISA	International Society of Automation
ISM	Industrial, Scientific and Medical
ISR	Interrupt Service Routine

LBR	Load Balancing Router
LLF	Least Laxity First
LOS	Line-of-sight
LWB	Low-power Wireless Bus
MA	Multiple Controller Activation
MAC	Media Access Control
MCU	Microcontroller Unit
MIMO	Multiple Inputs and Multiple Outputs
MEMS	Micro-electro-mechanical Systems
MPDU	MAC Protocol Data Unit
NACK	Negative Acknowledgement
NLOS	Non-line-of-sight
PDM	Proportional Deadline Monotonic
PIP	Packets in Pipe
PLC	Programmable Logic Controller
PRR	Packet Reception Rate
PSDU	PHY Service Data Unit
QoS	Quality of Service
RAM	Random Access Memory
RCRT	Rate-Controlled Reliable Transport Protocol
RM	Rate Monotonic
RS	Reed-Solomon
RSSI	Received Signal Strength Indicator
SA	Single Controller Activation
SER	Symbol Error Rate
SFD	Start Frame Delimiter
SINR	Signal to Interference plus Noise Ratio
SISO	Single Input and Single Output
SNR	Signal-to-noise Ratio

STSR	Single Transmitter Single Receiver
TCMC	Tree Convergecast Scheduling with Multiple Channels
TDMA	Time Division Multiple Access
UDP	User Datagram Protocol
UWB	Ultra-wideband
VCO	Voltage Controlled Oscillator
WSN	Wireless Sensor Network
WRAP	Wireless Reliable Acquisition Protocol

A

CURRICULUM VITÆ

PERSONAL DETAILS

Name	Dingwen Yuan
Date of Birth	23 November 1981
Place of Birth	Shanghai, China
Nationality	Chinese

EDUCATION

since 05/2010	Technische Universität Darmstadt, Darmstadt, Germany Doctoral candidate at Secure Mobile Networking Lab (SEEMOO), Department of Computer Science
10/2007 – 01/2010	Technische Universität Darmstadt, Darmstadt, Germany Information System Technology, Department of Electrical Engineering and Information Technology Degree: Master of Science (M. Sc.)
09/1998 – 07/2002	Shanghai Jiao Tong University, Shanghai, China Department of Automation, School of Electronics and Information Technology Degree: Bachelor of Engineering (B. Eng.)
09/1995 – 07/1998	Shanghai Shibei High School, Shanghai, China

WORK EXPERIENCE

since 05/2010	Technische Universität Darmstadt, Darmstadt, Germany Research associate at the Secure Mobile Networking Lab
07/2002 – 09/2007	Software engineer at Intelligent Traffic System (ITS) Department, Shanghai Electrical Apparatus Research Institute (SEARI), Shanghai, China

Darmstadt, 10. December 2015

B

AUTHOR'S PUBLICATIONS

PRIMARY AUTHOR

1. Dingwen Yuan, Matthias Hollick. TDMA Scheduling in Wireless Sensor Networks for Periodic Control Systems. Under submission.

2. Dingwen Yuan, Salil S. Kanhere, Matthias Hollick. Instrumenting Wireless Sensor Networks – A Survey On The Metrics That Matter. Under submission.

3. Dingwen Yuan, Matthias Hollick. Competition: Sparkle – Energy Efficient, Reliable, Ultra-low Latency Communication in Wireless Control Networks (Extended Abstract for the Dependability Competition). In *Proceedings of the 13th European Conference on Wireless Sensor Networks (EWSN)*, 2016

4. Dingwen Yuan, Matthias Hollick. Ripple. High-throughput, Reliable and Energy-efficient Network Flooding in Wireless Sensor Networks. In *Proceedings of the 16th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2015

5. Dingwen Yuan, Michael Riecker, Matthias Hollick. Making 'Glossy' Networks Sparkle: Exploiting Concurrent Transmissions for Energy Efficient, Reliable, Ultra-low Latency Communication in Wireless Control Networks. In *Proceedings of the 11th European Conference on Wireless Sensor Networks (EWSN)*, 2014

6. Dingwen Yuan, Matthias Hollick. Let's Talk Together: Understanding Concurrent Transmission in Wireless Sensor Networks. In *Proceedings of the 38th IEEE Conference on Local Computer Networks (LCN)*, 2013

7. Dingwen Yuan, Michael Riecker, Matthias Hollick. HOPSCOTCH: An adaptive and distributed channel hopping technique for interference avoidance in Wireless Sensor Networks. In *Proceedings of the 37th IEEE Conference on Local Computer Networks (LCN)*, 2012

8. Dingwen Yuan, Matthias Hollick. Optimization and Scheduling of Wireless Sensor Networks for Periodic Control Systems. In *Proceedings of the 11th GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze" (FGSN)*, 2012

9. Dingwen Yuan, Matthias Hollick. Tree-based Multi-channel Convergecast in Wireless Sensor Networks. In *Proceedings of the 13th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2012

CO-AUTHOR

1. Michael Riecker, Dingwen Yuan, Rachid El Bansarkhani, Matthias Hollick. Patrolling Wireless Sensor Networks: Randomized Intrusion Detection. In *Proceed-*

ings of the 10th ACM International Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet), 2014

2. Michael Riecker, Rainer Thome, Dingwen Yuan, Matthias Hollick. A Secure Monitoring and Control System for Wireless Sensor Networks. In *Proceedings of the 37th IEEE Conference on Local Computer Networks (LCN)*, 2012

3. Andreas Sewe, Dingwen Yuan, Jan Sinschek, Mira Mezini. Headroom-based Pretenuring: Dynamically Pretenuring Objects that Live "Long Enough". In *Proceedings of the 8th International Conference on the Principles and Practice of Programming in Java (PPPJ)*, 2010.

C

ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG

Ich versichere hiermit, dass ich die vorliegende Dissertation selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmitteln verfasst habe. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch nicht zu Prüfungszwecken gedient.

Darmstadt, 10. Dezember, 2015

Dingwen Yuan, M.Sc.