January 2018

Real-Time Guarantees For Wireless Networked Sensing And Control

Yu Chen
Wayne State University, yuchen.wayne@gmail.com

REAL-TIME GUARANTEES FOR WIRELESS NETWORKED SENSING AND CONTROL

by

YU CHEN

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2018

MAJOR: COMPUTER SCIENCE

Approved By:

_____

Advisor Date

_____

_____

_____

_____

DEDICATION

To my wife, my parents, my parents-in-law, and my family.

ACKNOWLEDGEMENTS

I am very grateful to my advisor Dr. Hongwei Zhang for his generous guidance, support, and encouragement on conducting my Ph.D. research. His deep understanding of his research field, diligent work ethic, and determination to pursue intellectual beauty has set a perfect example for me both in my research and personal life. The most important thing I learnt from Dr. Zhang through my Ph.D. journey is how to identify the right problem to solve and how to tackle a problem I have never encountered before. I am also very grateful to have Dr. Nathan Fisher as my another advisor and one of my dissertation committee members. Whenever I have questions on my research, Dr. Fisher has been always there to discuss with me, especially those related to real-time scheduling. I am also fortunate to have Dr. Loren Schwiebert, Dr. Henguang Li, and Dr. Zichun Zhong as my committee members. Despite their obligation in their departments, Dr. Schwiebert and Dr. Li have offered me insightful suggestions on how to improve my writing and presentation. I want to specially thank Dr. Zhong for agreeing to serve in my committee in short notice. I also want to thank Dr. Schwiebert, Amber Dawkins, Lanita Stewart, Brandi Gonzales, Areej Salaymeh, and other staff in my department for providing all the necessity to make my life easier. Thank the Graduate School, especially Kristy Case, for her patience in helping me with all the paperwork and format checking.

I want to express my sincere appreciation to my research collaborators and friends for their kind help. Dr. Feng Lin, Dr. Le Yi Wang and Dr. George Yin have been working with us to develop control algorithms for wireless networked sensing and control. Dr. Patrick Gossman and Mr. Matthew Lessons selflessly provided us with all the IT support we needed to build our testbed. I also enjoyed working with my lab members at Wayne State University: Dr. Yuehua Wang, Dr. Xin Che, Dr. Qiao Xiang, Dr. Xiaohui Liu, Chuan Li, Qing Ai, Pengfei Ren, and Lin Wang. During my job searching, I got a lot of referrals and endorsement from my friends: Hai Jin, Dr. Bing Li (Google), Dr. Shufeng Huang (Cisco), Dr. Yunsheng Wang (Kettering University), Dr. Qingsong Wen (Alibaba), Xianqiong Deng,

Xueying Linghu (Amazon), Xuejiao Li (Tesla), and so on. Changxin Bai and Rui Chen kindly helped me prepare the paperwork and logistics. Thank you all!

I owe my deepest gratitude to my family, who stood by me unconditionally for so many years. My parents have been supportive to me in my whole life, no matter how hard life is to them. My father fell down and injured his head badly last year when I was catching on a deadline. To avoid distraction to me, they lied to me when he was in the hospital for over two months. As soon as he returned home, he resumed working to pay the medical bill. However, they always tell me "Don't worry". Last but not least, I want to thank my wife for her endless love and understanding. We can only see each other a few times each year due to long distance. Sometimes she complained, sometimes she cried, but she never left me behind. She always has faith on me. While my father was ill, my wife, my in-laws, and my families took my responsibility to take care of him. Without my family, I could not finish my dissertation at all. I will love and take care of them for the rest of my life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Wireless networks are increasingly being explored for mission-critical sensing and control in emerging domains. For instance, in connected and autonomous vehicle, information from roadside unit and other cars must be delivered to the car in order to perform path planning and motion control in a timely manner. If the transmission fails or if the packets are not delivered on time, a car accident may well happen. For Industrial 4.0, reliable and timely communication between distributed sensors and controller plays a crucial role for the control decision. Out-of-date information will often be irrelevant and even leads to negative effects to the system. Therefore, the timely detection, processing, and delivery of information are often indispensable requirements in wireless networked sensing and control (WSC). However, WSC differs dramatically from the traditional real-time (RT) systems due to its wireless nature. New design objective and perspective are necessary to achieve real-time guarantees.

1.1 Our Contributions

Towards providing guaranteed reliability and timeliness for wireless sensing and control, we have made the following contributions.

- **Optimal Node Activation Multiple Access (ONAMA) scheduling protocol.** As the first step, we addressed the problem of predictable reliability (in terms of packets delivery ratio). We proposed the ONAMA protocol that schedules maximal number of concurrent transmissions while ensuring reliability. ONAMA has two pillars: a distributed MIS algorithm tailored to wireless characteristics and the pipelined precomputation technique to reduce DMIS's long delay. Extensive experiments on two testbeds, each with 127+ nodes, have independently shown that it increases concurrency by a factor of 3.7, increases throughput by a factor of 3.0, and reduces delay by a factor of 5.3 compared to the state of the art, while still catering to links' reliability requirements. .

- **Optimal request clustering algorithms for heterogeneous link reliability requirements.** In previous work, we assumed that each link only has one reliability requirement. In practice, however, each link may need to support various applications that require heterogeneous link reliability. One challenge for resource-constraint embedded devices is that they may only be able to maintain a limited number of requirements. Then, the problem is how to map heterogeneous reliability requirements into a limit set of service levels. We proposed two algorithms and formally proved their optimality.
- **A framework for probabilistic per-packet real-time guarantees.** After the predictable link reliability can be ensured, we continued to pursue the timeliness of transmissions. A probabilistic framework for per-packet real-time wireless communication guarantees was therefore proposed. Our notion of *real-time* differs fundamentally from existing work in the sense that it ensures in an *arbitrarily* long execution history, a *randomly* selected packet will meet its deadline in a user-specified probability. By R3 mapping, the upper-layer requirement and the lower-layer link reliability are translated into the number of transmission opportunities needed. By optimal real-time communication scheduling as well as admission test and traffic periods optimization, the system utilization is maximized while the schedulability is maintained. The proposed admission test is proved to be both sufficient and necessary. The simulation results show that the proposed scheme can meet the probabilistic real-time communication requirement.
- **Wireless probabilistic per-packet real-time guarantees with jitter control.** Message transmissions in wireless network may not only experience delay but delay variations (i.e., jitter) due to various factors such as scheduling algorithm, online traffic-load changes, channel fading, and environment dynamics. Besides meeting the deadlines of packets delivery, smaller delay jitter is often favorable since it means more stable transmissions. To achieve this goal, we proposed a two-level solution: the offline component explores the extra capacity by optimizing the deadlines, while the online

component further reduces the jitter by carefully considering the transmission order between different links.

1.2 Organization

The rest of this dissertation is organized as follows. In Chapter 2 we present the ONAMA protocol and experimental results. In Chapter 3, we propose two algorithms for optimal request clustering to guarantee heterogeneous link reliability requirements. A probabilistic framework for per-packet real-time wireless communication guarantees is then proposed in Chapter 4. We further investigate the problem of jitter control while ensuring deadlines in Chapter 5. We discuss the research plan and conclude the dissertation in Chapter 6.

CHAPTER 2: A MAXIMAL CONCURRENCY AND LOW LATENCY DISTRIBUTED SCHEDULING PROTOCOL FOR WIRELESS SENSOR NETWORKS

2.1 Introduction

Due to the broadcast nature of wireless communication, access to the shared wireless channel has to be coordinated to avoid interference. There are generally two ways to achieve this, contention-based [10][11][12][14] and TDMA-based [6][8][9][13].Contention-based protocols are easy to realize, but their performance can be highly dynamic and unpredictable owing to collision and random backoff, especially when node density is high and traffic load is heavy. This explains the prevalence of TDMA-based protocols in scenarios where quality of service (e.g., high throughput, high reliability, and low delay) has to be provided. For instance, the WirelessHART [19] and ISA SP100.11a [18] standards defined for industrial monitoring and control are both TDMA-based.

Conflict graph has been used to model wireless interference between neighboring nodes [7]. In a conflict graph, a node represents a contention entity. It can be either a node or a transmission link in the original network. A link between two nodes exists if data transmissions of the two represented contention entities interfere with each other according to an interference model (e.g., the Physical-Ratio-K model [36]). Once we acquire scheduling for a conflict graph, it is straightforward to translate it to the corresponding wireless network. From now on, graph is synonymous with conflict graph.

To avoid collision, a series of protocols have been proposed to schedule concurrent transmissions conflict-free in multi-hop wireless networks. In the Node Activation Multiple Access protocol (NAMA)[6], a node only accesses the channel if its priority is higher than all of its conflicting neighbors', where the priority is a hash function of its unique ID and the current time slot. While NAMA ensures no two nodes transmit simultaneously if they interfere with each other, it can lead to severe concurrency loss and channel underutilization.

This work is supported in part by NSF awards CNS-1136007, CNS- 1054634, GENI-1890, and GENI-1633.

Figure 2.1 illustrates such an example. In the conflict graph below, the number inside each node represents its priority. There is a link between two nodes if their transmissions collide. According to NAMA, only node with priority 7 can be active although nodes with priority 1, 2, 3, and 4 can transmit as well without causing collision.

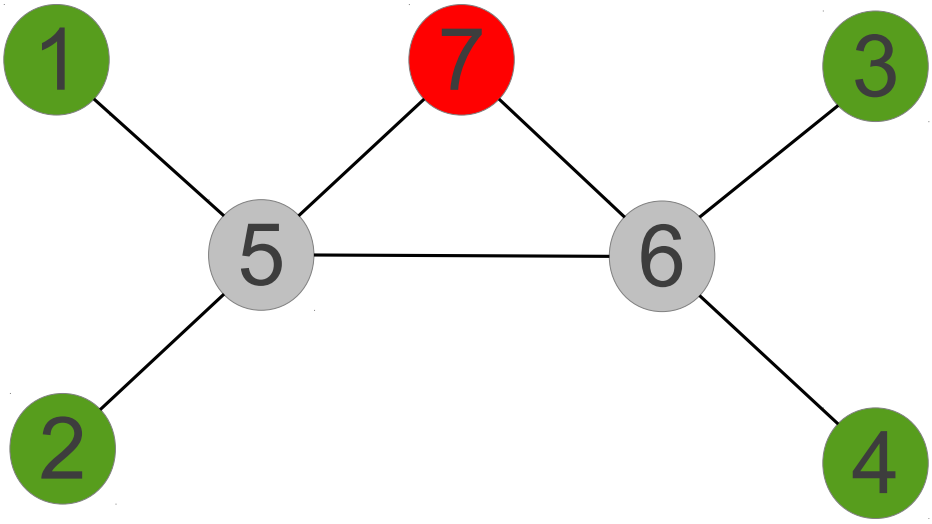


Figure 2.1: NAMA's concurrency loss.

Given the growing spectrum deficit resulted from ever-increasing demands and fixed amount of spectrum, we aim to squeeze the most capacity out of the limited spectrum. And we propose the Optimal Node Activation Multiple Access (ONAMA) scheduling protocol that activates as many nodes as possible while ensuring collision-free scheduling. It formulates the scheduling problem into finding a maximal independent set in the conflict graph. In graph theory, an independent set is a set of nodes in a graph, none of which are adjacent. An independent set is maximal if adding any other node makes it no longer an independent set. ONAMA includes the distributed maximal independent set (DMIS) algorithm, which identifies a maximal independent set (MIS) of a graph given all node priorities, calculated the same way as NAMA does.

NAMA can compute the schedule for each slot on the fly because it requires no packet exchange. By contrast, it takes multiple rounds of packet exchange for DMIS to find the schedule (i.e., MIS) for a slot. What's worse, the wireless channel is susceptible to packet

loss. If ONAMA also computes the schedule on the fly, it introduces significant delays for data delivery, especially in large networks. To reduce delay incurred by MIS computation while activating as many nodes as possible, we decouple the computation of MIS from data transmission by precomputing it in advance. When a certain slot comes, ONAMA simply looks up the precomputed MIS on the fly and activates a node if and only if it is in the MIS. To further reduce delay, we organize the precomputation of MISs for consecutive slots in a pipeline.

Contributions of this paper. (1) We develop a distributed scheduling protocol ONAMA that attains maximal activation in a multi-hop wireless network while causing no collision, even if the network is dynamic. It greatly enhances channel spatial reuse compared to the state of the art. (2) We devise a novel technique called pipelined precomputation to address the excessive delay in basic ONAMA. (3) We implement ONAMA on extremely resource-constrained TelosB [2] motes using TinyOS [3]. Evaluation in two independent testbeds has verified it increases concurrency by a factor of 3.7, increases throughput by a factor of 3.0, and reduces delay by a factor of 5.3 compared to existing work while maintaining reliability.

Organization of this paper. Section 2.2 details the ONAMA protocol. We evaluate it comprehensively in Section 2.3. Section 3.6 discusses related work. We wrap up the article by drawing some conclusions and pointing out some future directions.

2.2 The ONAMA Protocol

We elaborate on the design and implementation of the ONAMA protocol in this section. First we introduce the baseline version of ONAMA, namely the distributed MIS (DMIS) algorithm, to compute the MIS of a graph in a fully decentralized way. Next, we reduce DMIS's excessive delay by a novel approach called pipelined precomputation. Finally we tackle some challenges in implementing ONAMA on resource-scarce embedded devices.

We assume time on all nodes is synchronized and divided into slots. This can be attained by, for example, running a time synchronization protocol [37]. We also assume each node has a unique ID.

2.2.1 Distributed MIS

Inspired by the observation that NAMA can severely underutilize the channel, we decide to activate as many nodes as possible while still ensuring no two neighboring nodes are active together. In other words, we activate a maximal independent set of a graph in each slot through the following distributed MIS algorithm. Even though it resembles some other existing distributed MIS algorithms in appearance, especially the FastMISv2 algorithm [5], it is essentially different from them to factor in the peculiarities of wireless sensor networks, i.e., limited bandwidth, memory, and computational power.

In DMIS, a node stays in one of three states below at any given time:

- UNDECIDED: it has not decided whether to join the MIS or not.
- ACTIVE: it joins the MIS.
- INACTIVE: it does not join the MIS.

Initially, all nodes are UNDECIDED. In any slot t, each node computes the priority of itself and its neighbors according to Equation 2.1

$$p_i = Hash(i \oplus t) \oplus i. \tag{2.1}$$

i is the node id, $Hash(x)$ is a fast message digest generator that returns a random integer by hashing x, and p_i is i's priority. $\oplus$ concatenates its two operands. Note the second $\oplus$ is necessary to guarantee all nodes' priorities are distinct even when $Hash()$ returns the same number on different inputs. Based on the priority for each node, DMIS computes a MIS for slot t in multiple phases. Each phase consists of three steps:

1. Node v exchanges nodal states with its neighbors.
2. If node v's priority is higher than all its ACTIVE and UNDECIDED neighbors', it enters the MIS by marking itself ACTIVE; conversely, if any of its higher priority neighbors is ACTIVE, it marks itself INACTIVE.
3. Node v proceeds to the next phase only if its state is UNDECIDED.

When no node is UNDECIDED, the algorithm terminates. Theorem 2.2.1 shows DMIS does terminate in finite phases.

Theorem 2.2.1. *DMIS terminates in finite phases.*

Proof. Given a graph G and distinct priorities for all its nodes, in the first phase, there are a set of nodes $\mathbb{A}$ going from UNDECIDED to ACTIVE because their priorities are higher than all of their neighbors'. In the following phase, all neighbors of $\mathbb{A}$, denoted as $\mathbb{I}$, become INACTIVE. Removing all nodes in $\{\mathbb{A} \cup \mathbb{I}\}$ and their adjacent links from G, we denote the resulting subgraph as G' and the node set of G' as $V(G')$. No node in $\mathbb{A}$ is adjacent with any node in $V(G')$ because otherwise that node in $V(G')$ would be INACTIVE and in $\mathbb{I}$. So removing all nodes in $\mathbb{A}$ and their adjacent links from G does not affect the ensuing phases. Similarly, removing all nodes in $\mathbb{I}$ and their adjacent links from G does not affect neither since a node becomes ACTIVE or INACTIVE irrespective of its INACTIVE neighbors in phase 2. After the first two phases, we only have to run DMIS on the residual subgraph G'. After two more phases, G' becomes G''. This process continues till the residual subgraph is empty. Because some nodes are removed every time we go from a graph to its subgraph and there are finite nodes in a graph, DMIS terminates in finite phases. □

Analogous to complexity analysis in [5], it's easy to prove that DMIS terminates in $O(\log(n))$ phases on average, where n is the number of nodes in the graph. Interested readers can refer to [5] for more details of the proof. Theorem 2.2.2 shows DMIS finds a MIS of the graph.

Theorem 2.2.2. *The set $\mathbb{S}$ of all ACTIVE nodes is a MIS of the graph after DMIS terminates.*

Proof. We prove the theorem in two steps.

1. $\mathbb{S}$ is an independent set.

   We prove this by contradiction. Suppose $\mathbb{S}$ is not an independent set, then there exist two adjacent nodes u and v in $\mathbb{S}$. Since a node only becomes ACTIVE if its priority is higher than all its ACTIVE and UNDECIDED neighbors' and u is ACTIVE, $p_u > p_v$. Likewise, $p_v > p_u$. Contradiction. Thus $\mathbb{S}$ is an independent set.

2. The independent set $\mathbb{S}$ is maximal.

   When DMIS terminates, a node is either ACTIVE or INACTIVE. $\forall u \notin \mathbb{S}$, it is INACTIVE, which means there exists an ACTIVE neighbor $v \in \mathbb{S}$, whose priority is higher than u's. $\{u \cup \mathbb{S}\}$ is not independent since u and v are adjacent. Hence $\mathbb{S}$ is a MIS.

□

The resulting MIS is the set containing all ACTIVE nodes, which are to be activated in slot t. It is noteworthy that exchanged nodal state does not include priority, which is computed locally even for neighbors'.

2.2.2 Pipelined precomputation

One salient feature of NAMA is that it requires no packet exchange to compute a schedule and can thus be called on the fly. In a TDMA setting, a node can call NAMA as a subroutine at the beginning of a slot to instantaneously determine if it should be active in that slot. The idea of doing the same for DMIS is enticing, but the ability to do so proves elusive. This is because, unlike NAMA, running DMIS requires multiple phases and each phase incurs significant delay mainly due to contention to access the shared channel and unreliable channel in step 1. The resulting excessive delay for data delivery is detrimental for a wide range of time-sensitive applications.

To reduce DMIS's delay while retaining its high concurrency, we decouple it from data transmission by precomputing MIS of a slot M slots in advance. M is chosen such that DMIS converges within M slots, at least with high probability. Ideally, M is set as the product of the number of phases it takes for DMIS to complete and the number of slots needed for every node to exchanges its nodal status with all its neighbors in each phase. To determine the former, we log the conflict graphs during runtime and execute DMIS over them offline to see how many phases needed for its completion. The latter depends on the control signaling scheme and qualities of links between a node and its neighbors. In reality, M is also bounded by available memory as we will elaborate in subsection 2.2.4.

In slot t, DMIS starts computing MIS for future slot $(t + M)$ using nodes' priorities at slot $(t + M)$. The intermediate result, i.e., the current MIS, is stored till slot $(t + M)$. When time reaches slot $(t + M)$, a node simply looks up the precomputed MIS and decides to become active or silent, without computing it on the fly like in NAMA. Since it takes M slots to compute the MIS for each slot, in slot t, MISs for slot $(t + 1), (t + 2), ..., (t + M)$ are being computed. We organize their computation into a pipeline, where the MIS computation of consecutive M slots overlaps. This is more efficient than computing them sequentially. Moreover, we aggregate a vector of M states and exchange them in a single control packet at once, other than convey each of the M states using a separate packet. This greatly saves channel resources.

Figure 2.2 shows an example of the proposed precomputed pipeline in action when M is 4. The x-axis denotes time in slot, the y-axis denotes the slot whose MIS is being computed. The computation of MIS for slot 4 starts at slot 0 and continues in slots 1, 2, and 3. In slot 4, MIS for this slot has been precomputed and is ready for immediate activation. Similarly, MIS for slot 5 is ready at slot 5, MIS for slot 6 is ready at slot 6, and so on. In slot 3, MISs for the upcoming slots 4, 5, 6, and 7 are being computed simultaneously.

2.2.3 Dynamic graph

A graph changes over time as nodes join or leave the network, links establish or break. This change confuses DMIS because it assumes the graph remains static before it converges. We solve this issue by a snapshot-base approach. Specifically, when starting to compute the MIS for a future slot $(t + M)$ in slot t, we take a snapshot of the graph and use it for the remaining computation even the graph changes within M slots. Hence, the graph is consistent for each call of DMIS. One potential side effect of this approach is that ONAMA defers the usage of the latest graph, which may degrade application performance at upper layer. Even if this is the case, the degradation can be mitigated by making M smaller as we shall discuss in subsection 2.2.4.

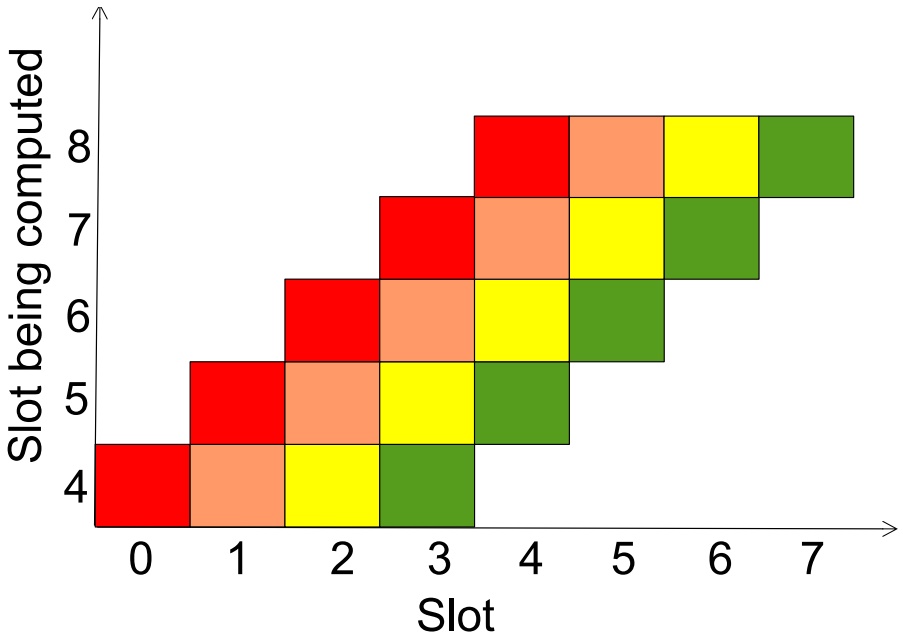


Figure 2.2: Pipelined precomputation.

Putting all building blocks together, we present the ONAMA protocol in Algorithm 1 as follows:

2.2.4 Implementation issues

A typical embedded device is equipped with extremely limited memory. For instance, a TelosB [2] mote has only 10 KB of RAM. Implementing ONAMA on such resource-constrained devices poses an additional challenge that is not found on resource-rich ones. To overcome this challenge, we expose several key parameters for fine tuning to let upper layer trade off between memory usage and performance. There are two places in ONAMA that can expend significant amount of memory, especially when the network is large.

First, each node maintains a table containing all its potential neighbors with size L. To store graph snapshot for each slot, a node needs 1 bit for each node in its neighbor table, indicating whether they interfere or not. It thus costs each node $L * M$ bits to store local graph snapshots. To reduce snapshot footprint, we take snapshots every other G slots, instead of every slot. After each snapshot, DMIS uses it to compute the MIS of the next

Algorithm 1 Pipelined precomputation for M slots ahead

```
1: /* Pipelined precomputation for M slots ahead */
2: for i ∈ [t + 1, t + M]
3:     call DMIS to precompute state for future slot i;
4: end for
5: Procedure DMIS
6: /* Keep on computing my state until it is decided */
7: while my state is UNDECIDED
8:     Exchange nodal states with my neighbors;
9:     Compute priorities for myself and my neighbors using Eq. 1;
10:    if my priority is higher than all ACTIVE and UNDECIDED neighbors
11:        Set my state as ACTIVE;
12:    else if any of my higher priority neighbors is ACTIVE
13:        Set my state as INACTIVE;
14:    end if
15: end while
```

G consecutive slots. This reduces snapshot footprint by a factor of G. Nevertheless, larger G is not always desirable since it makes the protocol less agile to graph change. G can be tuned to strike a balance between memory consumption and protocol agility.

Second, in step 1 of a phase in DMIS, a node exchanges states with neighbors by sending and receiving control packets, which can piggyback upper layer payload as well if space permits. We further divide each slot into S subslots, out of which one is reserved for data packets and the rest for control packets. Only one data or control packet can be transmitted by a node in each subslot. In total, each node stores $L * M$ intermediate states for pipelined precomputation. On the one hand, because a fixed number of control packets are needed for the convergence of DMIS for a slot on a given graph, a larger S packs more control packets into a slot and thus lessens M and memory expenditure. On the other hand, a larger S also increases control overhead and lowers channel utilization for data delivery. A judicious selection of S again depends on memory consumption and performance tradeoff.

2.3 Evaluation

2.3.1 Methodology

We test ONAMA as a component of the PRK-based scheduling (PRKS) protocol. PRKS [37] is a TDMA-based distributed protocol to enable predictable link reliability based on the Physical-Ratio-K (PRK) interference model [36]. The PRK model marries the amenability to distributed protocol design of the ratio-K model (i.e., interference range = K * communication range) and the high fidelity of the signal-to-interference-plus-noise ratio (SINR) model. Through a control-theoretic approach, PRKS instantiates the PRK model parameters according to in-situ network and environment conditions so that each link meets its reliability requirement after convergence. As stated in Section 2.2, PRKS essentially defines a conflict graph for a given wireless network: a node in the graph denotes a link with data transfer in the network, and a link exists between two nodes in the graph if the corresponding links in the network interference with each other according to the PRK model and its instantiated parameters. It is noteworthy that even the underlying network is (semi-)static, the conflict graph is dynamic due to the constantly changing PRK model parameters. Under the condition that link reliability is ensured, PRKS schedules as many nodes as possible in the conflict graph, which is exactly what ONAMA intends to do. Besides ONAMA, PRKS contains many other components such as link estimator, controller, forwarder, time synchronization, and logging. The runtime interactions and resource sharing between ONAMA and them can cause undesirable effects that do not manifest when running ONAMA in isolation. By integrating ONAMA as a part of complex applications like PRKS, we can test the robustness of its design and implementation under dynamic conflict graphs.

To demonstrate the benefits of ONAMA, we compare the following two PRKS variants:

- PRKS-NAMA: PRKS running on top of NAMA.
- PRKS-ONAMA: PRKS running on top of ONAMA.

At the beginning of a time slot, every node calls the NAMA or ONAMA component to decide whether any incident links shall be active in this slot. We measure their performances in two sensor network testbeds, NetEye [4] and Indriya [1]. Indriya is a wireless sensor network deployed across three floors of the School of Computing at the National University of Singapore, while NetEye is a wireless sensor network testbed on the third floor of the Department of Computer Science at Wayne State University. In both testbeds, we set M, G, and S to be 112, 16, and 10, respectively.

Network and traffic settings. In NetEye and Indriya, we choose each node with probability 0.8 and 0.5, respectively, and the resulting set of nodes forming a random network. NetEye deploys 130 TelosB motes in a grid with every two closest neighboring motes separated by 2 feet. The random network is generated by using a subset of the 130 motes in experiment, in particular, using each node in NetEye with a probability of 0.8. Similarly, nodes in Indriya are chosen with a probability of 0.5. For each chosen node A, another node B (also in the random network) is chosen such that the packet delivery ratio (PDR) of the link from A to B is at least 95% in the absence of interference. For each link, the sender transmits a 128-byte packet to the receiver every 20 ms. Data transmission power is fixed at -25dBm, i.e., power level 3 in TinyOS.

2.3.2 Measurement results

NetEye testbed. For various PDR requirements, Figure 2.3 shows the mean concurrency (i.e., number of concurrent transmissions in a time slot) of PRKS-NAMA and PRKS-ONAMA, as well as a state-of-the-art centralized scheduling protocol iOrder [17], which also activates as many links as possible while ensuring each link meets its PDR requirement. Not only does PRKS-ONAMA significantly increase concurrency over PRKS-NAMA by up to 270%, but also it achieves a concurrency equal or close to that of iOrder despite its distributed nature. This clearly demonstrates the effectiveness of DMIS in ONAMA to activate more links simultaneously. It is worthwhile to note that though ONAMA seems to outperform iOder in some regions, they are the same statistically speaking.

Figures 2.4 and 2.5 show the boxplots of PDR in PRKS-NAMA and PRKS-ONAMA for different PDR requirements, respectively. We see even though PRKS-ONAMA improves concurrency and channel reuse enormously, it still guarantees that link PDRs meet requirements as PRKS-NAMA does. Because of higher concurrency without PDR sacrifice, PRKS-ONAMA's throughput is expected to be higher than PRKS-NAMA's as verified by Figure 2.6. Specifically, PRKS-ONAMA's throughput is 3.0, 2.9, 2.7, 2.5 times of PRKS-NAMA's when the PDR requirement is 70%, 80%, 90%, and 95%, respectively. We also note that as PDR requirement increases, throughputs in both protocols decrease due to lower concurrency. Figure 2.7 shows PRKS-ONAMA reduces the mean delay of PRKS-NAMA by a factor of 5.3, 4.6, 4.0, and 3.8 when the PDR requirement is 70%, 80%, 90%, and 95%, respectively, where the delay is defined as the difference between the packet transmission time and reception time. This significant improvement is due to both DMIS and pipelined precomputation in ONAMA. Specifically, with a larger concurrency from DMIS, a node has more transmission opportunities. Given a same PDR requirement, ONAMA has more chances to transmit than NAMA, thus delivers data faster. In addition, ONAMA's pipelined precomputation does not require computing transmission slot on-the-fly as NAMA does, it simply determines to be active or not according to the precomputed states.

Indriya testbed. Figures 2.8 and 2.9 show the link PDRs of PRKS-NAMA and PRKS-ONAMA under different PDR requirements in Indriya, both meeting their requirements as in NetEye.

Figures 2.10, 2.11, and 2.12 show the mean concurrency, mean throughput, and mean delay of both protocols under different PDR requirements in Indriya, respectively. We see similar relative performance as in NetEye but the degree of improvement is smaller. For instance, PRKS-ONAMA's throughput is only 1.5, 1.3, 1.3, 1.4 times of PRKS-NAMA's when the PDR requirement is 70%, 80%, 90%, and 95%. This is because links in Indriya are sparser and a larger portion of them are already activated using NAMA. It is also worthwhile

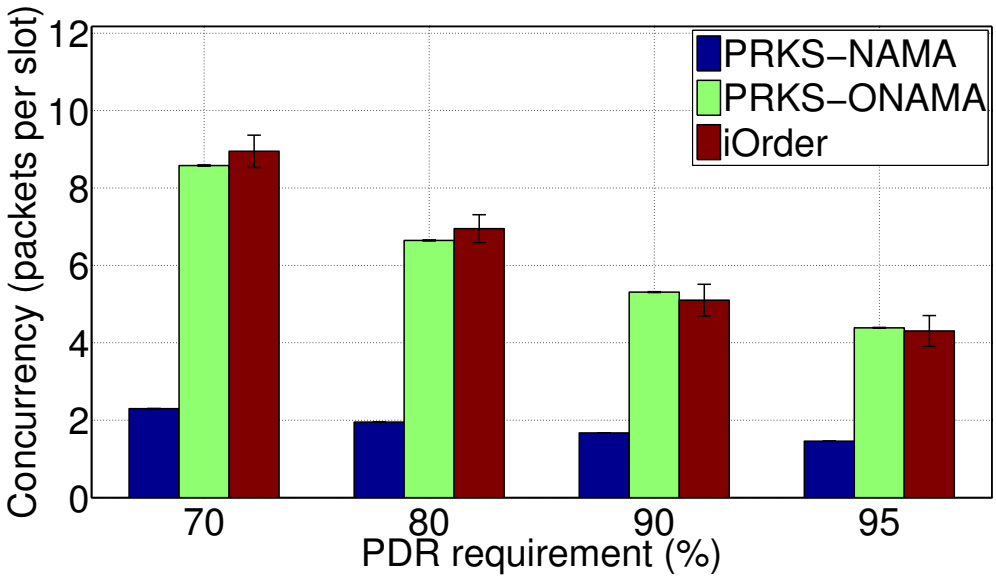


Figure 2.3: Mean concurrency in NetEye.

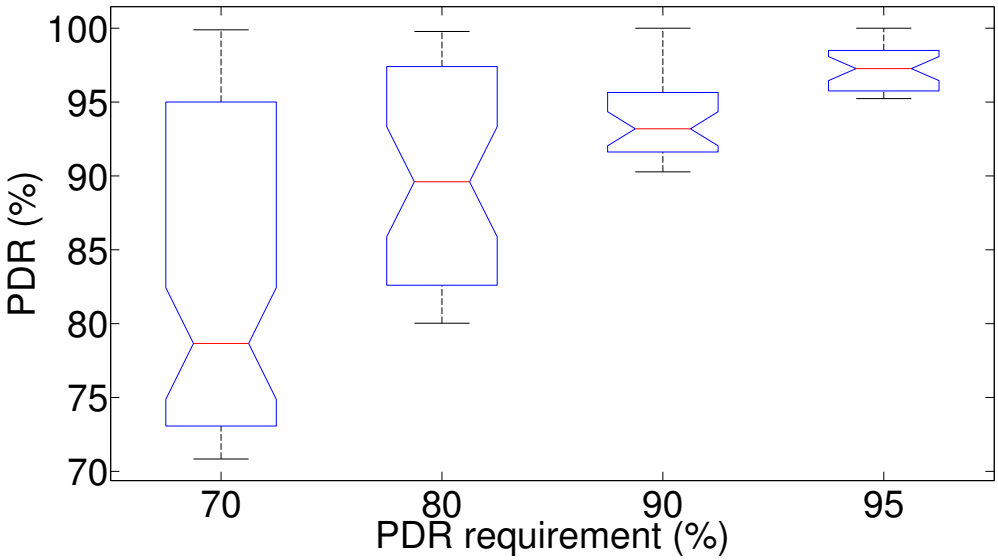


Figure 2.4: Packet delivery ratio (PDR) of PRKS-NAMA in NetEye.

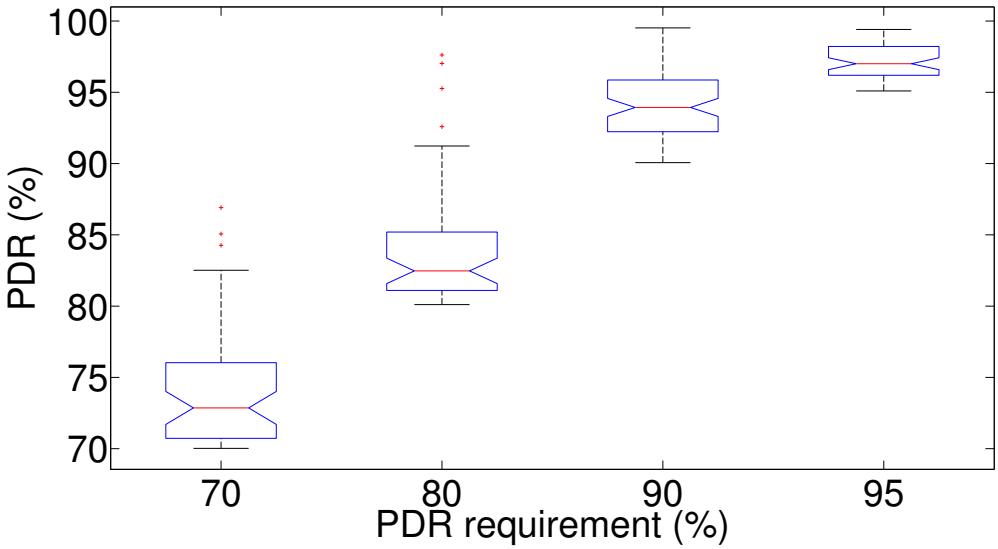


Figure 2.5: Packet delivery ratio (PDR) of PRKS-ONAMA in NetEye.

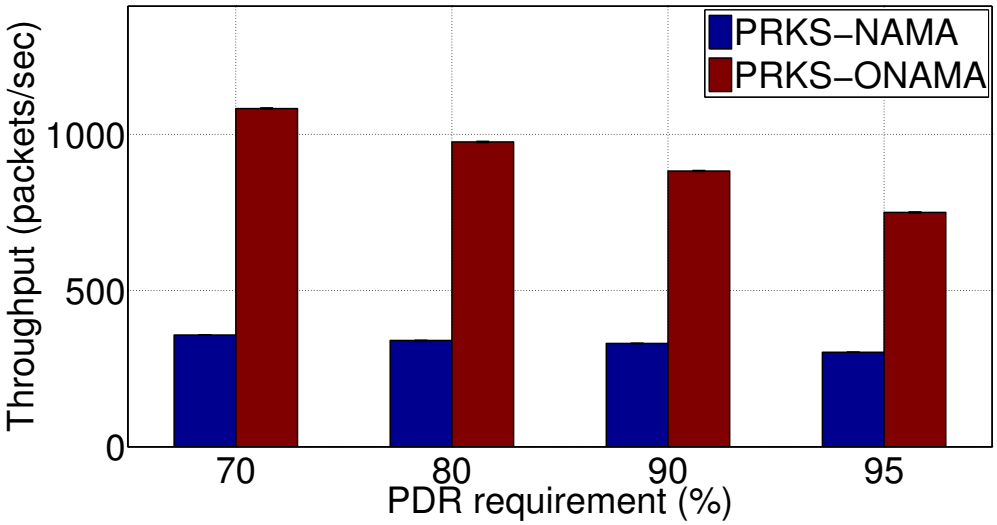


Figure 2.6: Mean throughput in NetEye.

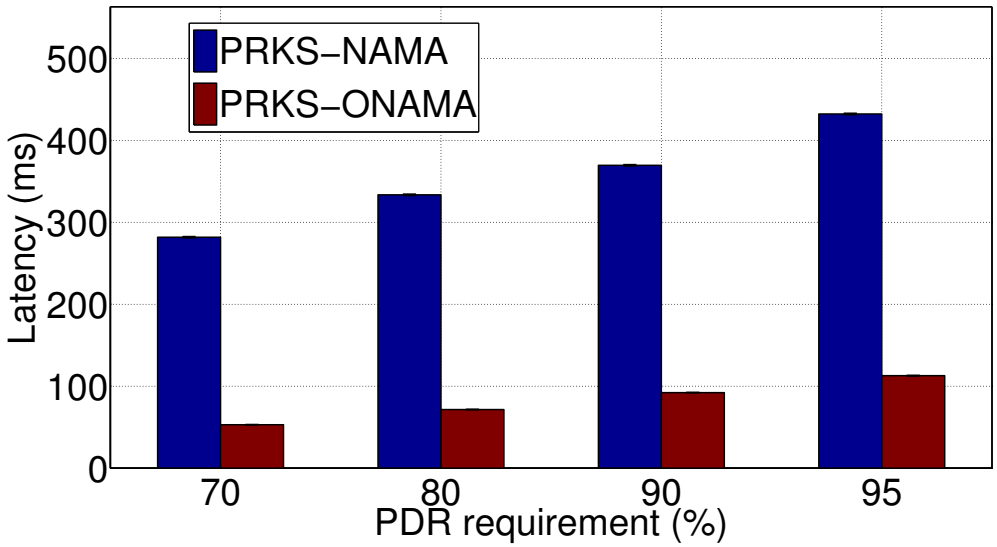


Figure 2.7: Mean delay in NetEye.

to note that though ONAMA seems to have larger concurrency than iOder in some regions of Figure 2.10, they are the same statistically speaking.

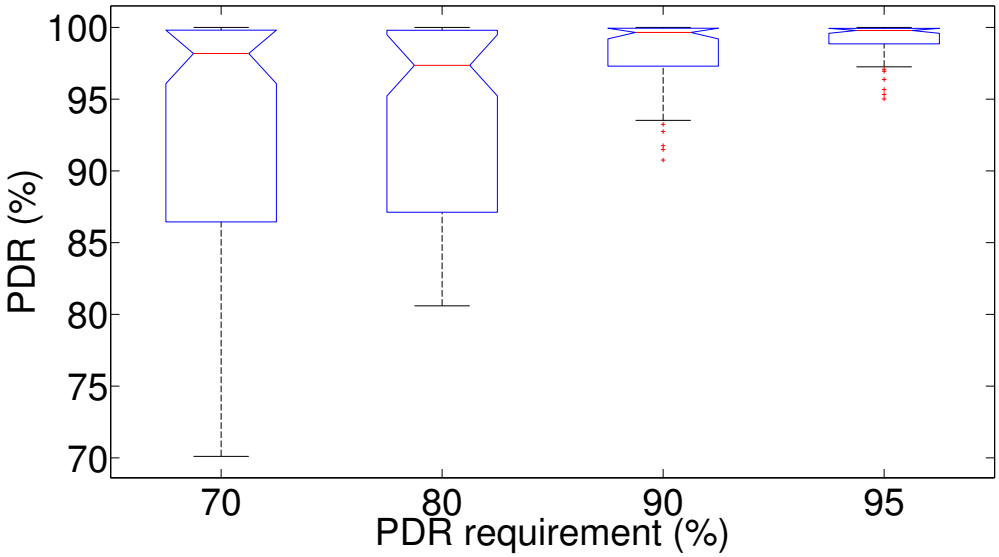


Figure 2.8: Packet delivery ratio (PDR) of PRKS-NAMA in Indriya.

2.4 Related Work

There are a plethora of protocols to schedule transmissions collision-free in ad hoc networks, such as NAMA and HAMA, to name a few. In Node Activation Multiple Access

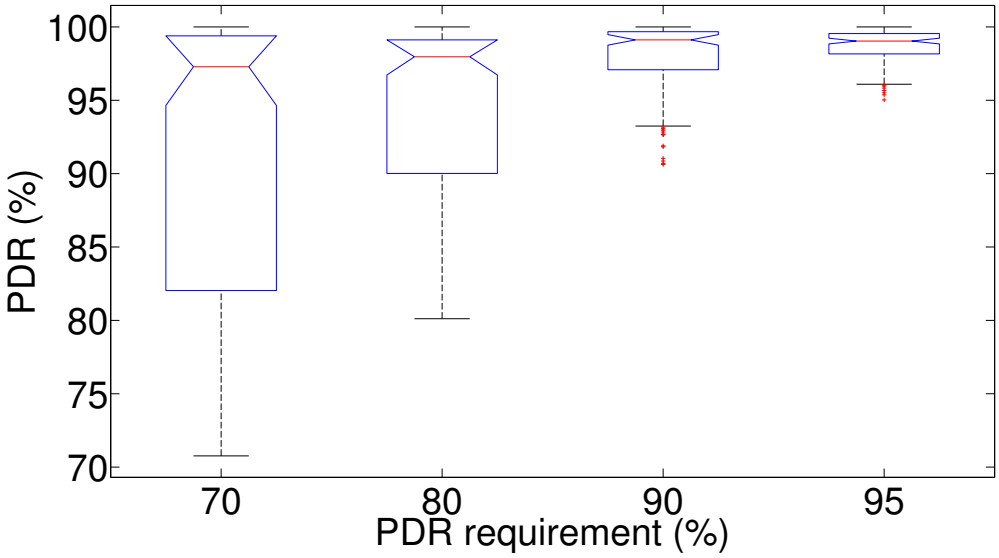


Figure 2.9: Packet delivery ratio (PDR) of PRKS-ONAMA in Indriya.

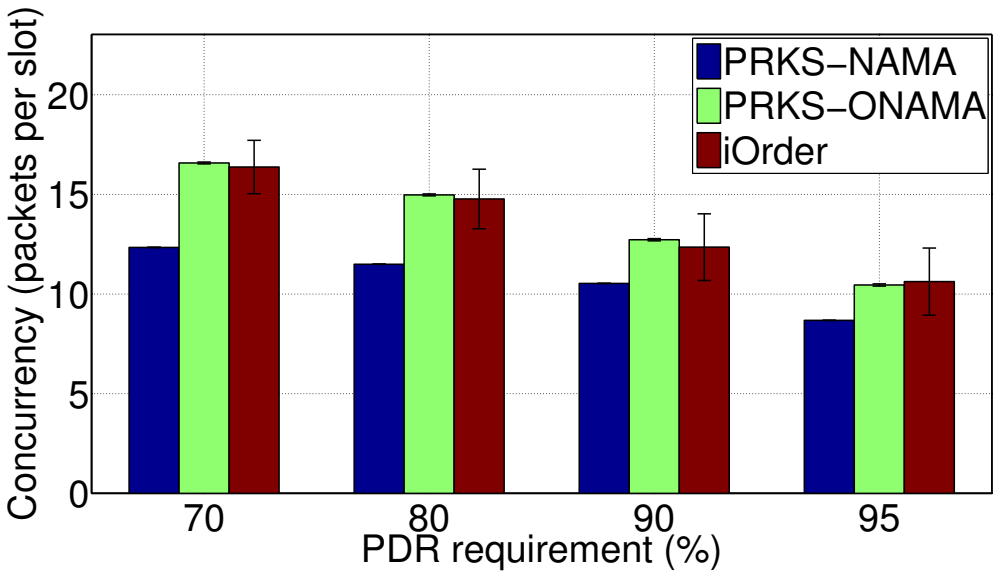


Figure 2.10: Mean concurrency in Indriya.

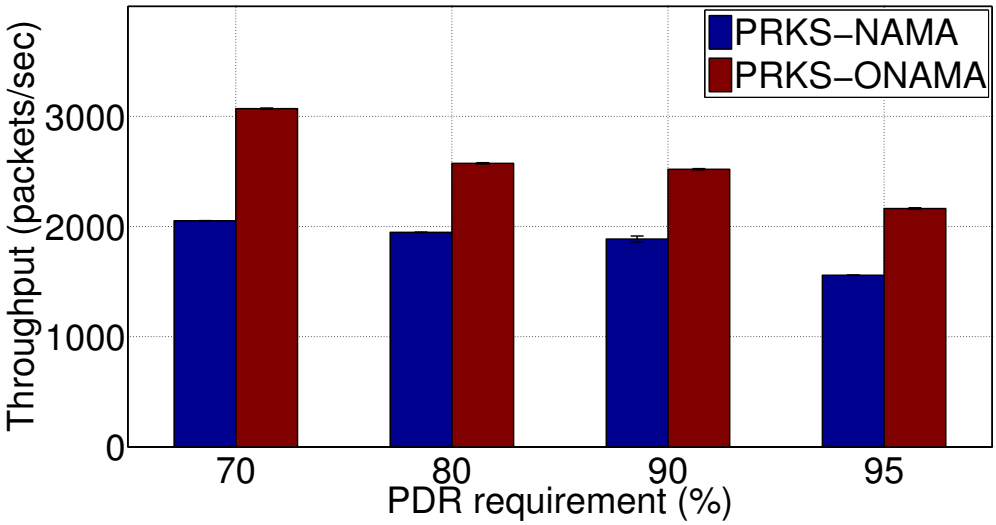


Figure 2.11: Mean network throughput in Indriya.

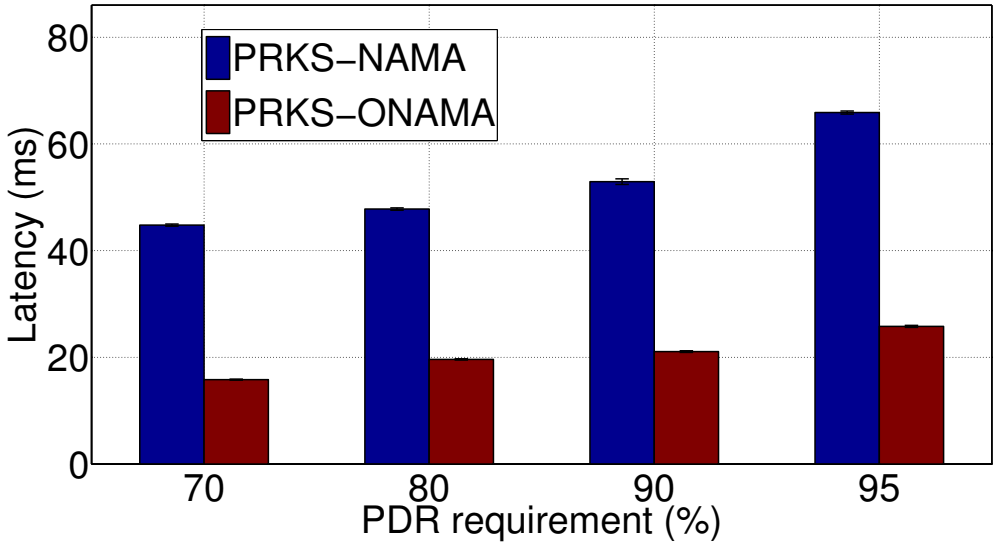


Figure 2.12: Mean delay in Indriya.

protocol (NAMA) [6], each node generates a unique priority by hashing and only accesses the channel if its priority is the higher than those of its one-hop and two-hop neighbors. It achieves collision-free scheduling, but suffers from low channel utilization as some nodes are not activated even their activations cause no collision. Hybrid Activation Multiple Access protocol (HAMA) [16] builds upon and improves NAMA by activating non-contending entities, both nodes and links, whenever possible, thus it utilizes channel more efficiently. Unfortunately, it requires radios capable of code division multiplexing. In contrast with HAMA, ONAMA fully utilizes channel without additional requirement on radios.

Orthogonally, plenty of work identifies MIS in a distributed fashion[14][15], with the FastMISv2 algorithm in [5] being the closest to our work. Akin to DMIS in ONAMA, FastMISv2 also runs in multiple phases; in each phase, FastMISv2 includes a node in the MIS only if its priority is higher than those of all its neighbors, which are excluded from the MIS. There are quite a few critical distinctions, though. (1) Priority is calculated from a pseudorandom number generator (PRNG) with generally different seeds on different nodes, not from a common hash function. A node obtains its neighbor's priority by packet exchange, unlike in DMIS where a node computes its neighbor's priority directly, which expends precious channel resources. Additionally, a new priority is generated from PRNG in each phase, which consumes large amount of MCU time on embedded devices, while DMIS only generates priority once before the first phase. (2) Designed for generic distributed settings, FastMISv2 ignores control signalling and doesn't address the unique challenge posed by unreliable wireless channel. By contrast, ONAMA tackles the challenge explicitly in control signalling by incorporating wireless characteristics. (3) FastMISv2 does not handle the long delay incurred by MIS computation, making it unsuitable for delay-sensitive applications. (4) Lastly but importantly, FastMISv2 assumes a static graph and fails if the graph changes over time.

Finally, unlike all the aforementioned existing work that is evaluated in simulation at best, ONAMA is implemented on resource-limited devices and verified over two real-world testbeds.

2.5 Summary

In this paper, we propose the ONAMA protocol that schedules maximal number of concurrent transmissions collision-free in a multi-hop wireless network. ONAMA has two pillars: a distributed MIS algorithm tailored to wireless characteristics and the pipelined precomputation technique to reduce DMIS's long delay. Extensive experiments on two testbeds, each with 127+ nodes, have independently shown that it increases concurrency by a factor of 3.7, increases throughput by a factor of 3.0, and reduces delay by a factor of 5.3 compared to the state of the art, while still catering to links' reliability requirements. Not limited to PRKS, ONAMA can be used as a primitive for other applications that require distributed MIS activation. Moreover, pipelined precomputation can be a general technique employed by other distributed applications, where some information is needed imminently but obtaining it requires considerable amount of time. One of the future directions is to investigate power control mechanisms to further improve the performance of ONAMA.

CHAPTER 3: OPTIMAL REQUEST CLUSTERING FOR LINK RELIABILITY GUARANTEE IN WIRELESS NETWORKED CONTROL

3.1 Introduction

Wireless networks are increasingly being explored for real-time control of physical processes [20] [21]. In wireless networked control systems such as those in industrial automation [22] [23], ensuring predictable communication link reliabilities (usually characterized by packet delivery ratios) among sensors, controllers, and actuators is critical. In such scenarios, the application layer of each sender requests a certain link reliability, and its lower layers try to deliver the associated data packets to the receiver at the requested reliability level. Providing reliability less than what is requested may cause control failures, while providing reliability higher than the actual need could lead to underutilization of communication resources.

To guarantee that packets are delivered at requested reliability levels, Zhang et al. proposed PRKS for wireless networked control [37]. PRKS is a TDMA-based distributed protocol to enable predictable link reliability based on the Physical-Ratio-K (PRK) interference model [36]. Through a control-theoretic approach, PRKS instantiates the PRK model parameters according to in-situ network and environment conditions so that each link meets its reliability requirement. In particular, PRKS defines a conflict graph for a given wireless network: a node in the graph denotes a link with data transfer in the network, and a link exists between two nodes in the graph if the corresponding links in the network interfere with each other according to the PRK model. Under the condition that link reliability is ensured, PRKS schedules as many nodes as possible in the conflict graph.

What remains open is how to guarantee heterogeneous reliability requirements for applications. PRKS has only considered the case when there is one reliability request for each link. In wireless networked control, however, different reliability requests from different applications need to be satisfied. For example, safety-critical information and non-safety-

critical information gathered at the sender may well need different reliability guarantees. Ideally, the lower layer should maintain an infinite number of reliability levels to support all types of applications. In practice, however, providing only a (small) set of quantized reliability levels makes sense in many respects. Firstly, it is difficult and costly to support infinite number of reliability levels, especially for resource-constrained embedded systems. Secondly, performance analysis is more tractable with finite number of reliability levels. Then an important question is as follows:

Given a set of reliability requests and the maximum number of service levels that can be maintained, how to determine a set of service levels, and how to map each request to an appropriate service level, such that each request is guaranteed and the total cost of mapping is minimized?

In this paper, we formally formulate the problem as an optimal request clustering problem in the sense that each service level acts as a cluster, while all the requests mapped into a same service level are said to be within a same cluster. Theoretically, a service level may pick any real number between 0 and 1 (i.e., packet delivery ratio is between 0 and 100%), making the problem intractable. Towards addressing the problem, we have made the following contributions:

- We formulate and propose an optimal solution for the Migratory Clustering Problem, where a request can migrate from one service level to another during its lifetime.
- We also formulate and propose an optimal solution for the Non-Migratory Clustering Problem, where a request must stay at the same service level once it is assigned to a certain level.
- Unlike most existing clustering algorithms that just minimize the clustering cost, our algorithms also ensure that each reliability request is clustered to a service level no less than it, thus the reliability is guaranteed.
- Our clustering algorithms take into account the lifetimes of requests. On the contrary, most existing clustering algorithms did not consider the lifetimes of requests, thus a

current optimal clustering may not be optimal in the future due to the expiration of some requests.

3.2 Migratory and Non-migratory Clustering Problem

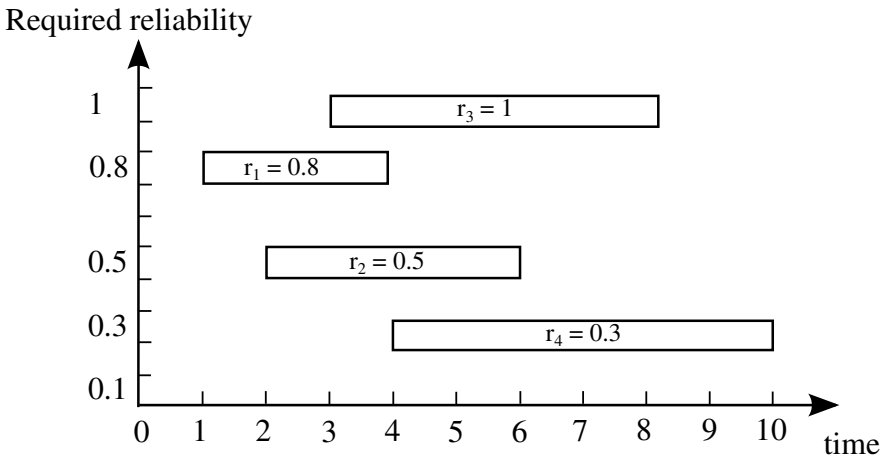
3.2.1 System model

We consider a transmitting node in a multi-hop wireless control network. The node may generate or forward data to next hop. Assume that time is slotted and indexed by $t \in \{0, 1, 2, ...\}$.

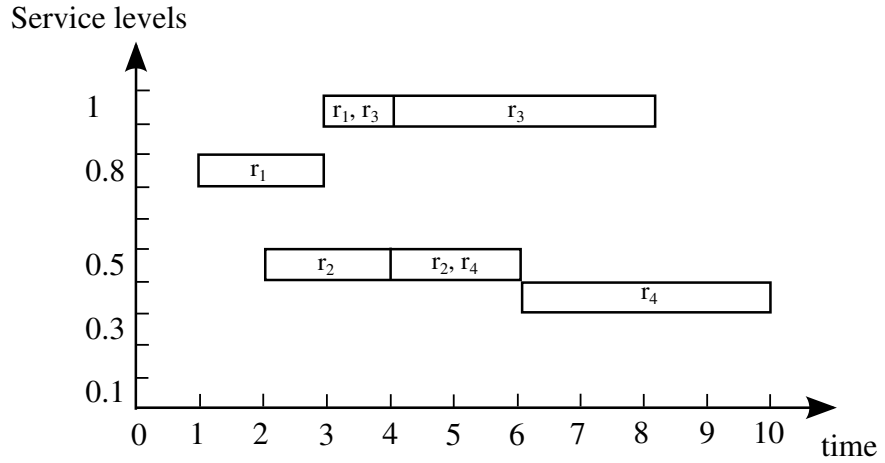
Request. A request $r_i(t)$ is characterized by a triple (r_i, a_i, e_i), where $r_i \in [0, 1]$ is the required reliability, a_i is the time when the request arrives at the node, and e_i is the time when the request expires, respectively. The required reliability stays unchanged during the lifetime of the request, i.e., $r_i(t) = r_i$ for $t \in [a_i, e_i]$. Figure 3.1(a) shows an example of 4 requests. In the example, request $r_1(t)$ arrives at $t = a_1 = 1$ and expires at $t = e_1 = 4$, requiring a reliability of $r_1 = 0.8$, and so on. Note that given a time instant t, the number of live requests gathered at the node may be less than 4. Unlike bursty data traffic in the regular Internet, data traffic in wireless networked control tend to be periodic (e.g., periodic sensor sampling) and predictable [41]. Thus we assume, as in many existing studies [32] [80], that the request information is known in this study.

Service level. A service level is the actual link reliability provided to a request. In a running system, it usually requires certain memory and computation overhead to maintain a service level [37]. It is therefore reasonable to assume that at any time t, the node can only maintain L service levels. Upon its arrival, each request must be assigned to a service level $s_j(t) \in [0, 1]$ ($j \in \{1, 2, ..., L\}$) immediately. A service level acts as a cluster and can host multiple requests. Without ambiguity, we will use the terms *service level* and *cluster* interchangeably in the paper.

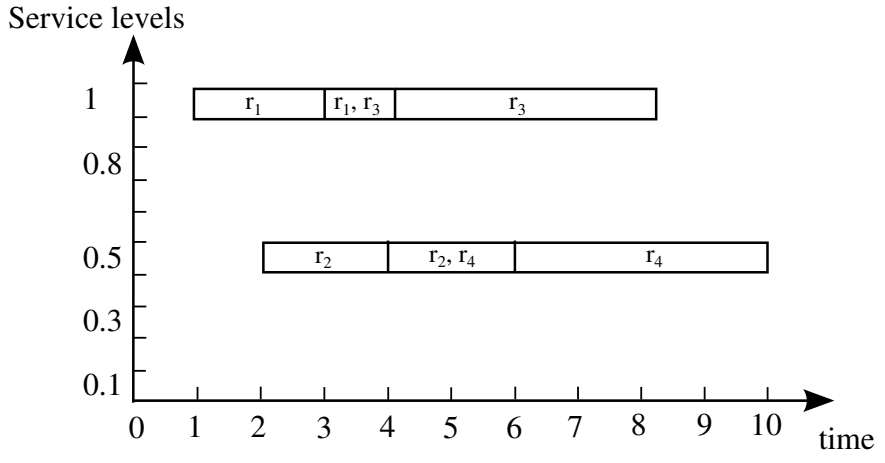
Migratory and non-migratory clustering. We consider two types of clustering: *migratory* and *non-migratory* clustering. The former allows a request to make a transition to another cluster from one time instant to another, while the latter requires a request to stay



(a) An example of 4 requests



(b) Migratory clustering for the requests shown in (a)



(c) Non-migratory clustering for the requests shown in (a)

Figure 3.1: An illustration of migratory and non-migratory clustering with 4 requests and 2 service levels. At any time t, only 2 service levels can be maintained.

in the same cluster once assigned. Figure 3.1(b) shows an example of migratory clustering for the requests shown in Figure 3.1(a). For instance, request 1 is assigned to service level 0.8 from $t = 1$ to $t = 3$, and assigned to service level 1 from $t = 3$ to $t = 4$. Figure 3.1(c) shows an example of non-migratory clustering for the same requests, where request 1 must stay at service level 1 during its lifetime due to the non-migratory constraint. In both cases, there are only two service levels maintained at any time. Generally, migratory clustering finds its application in scenarios where migrating a request from one cluster to another introduces negligible overhead. On the other hand, non-migratory clustering is more suitable for cases where migration is expensive or infeasible. Therefore, we define the clustering cost as time-dependent as follows:

$$\underset{r_i(t)\to s_j(t)}{c_i(t)} = \begin{cases} s_j(t) - r_i, & \text{if } s_j(t) \geq r_i; \\ +\infty, & \text{if } s_j(t) < r_i. \end{cases} \tag{3.1}$$

The notation $r_i(t) \to s_j(t)$ denotes that a request $r_i(t)$ is mapped to a service level $s_j(t)$ at time t. The definition is quite intuitive: if the provided reliability is higher than the request, the cost is the difference between the service level and the request, accounting for the overprovisioning of the resource; if a service level is lower than the request, the cost is infinite, meaning the reliability requirement cannot be met, thus the clustering is not acceptable. The hard constraint on reliability makes sense for wireless networked control since reliability lower than the request may well lead to control divergence or even catastrophe.

3.2.2 Problems definition

Migratory Clustering Problem (MCP).

Given a set of requests $\mathbf{r}(t) = \{r_1(t), r_2(t), ..., r_N(t)\}$. For any $i \in [1, N]$ and $t \in [a_i, e_i]$, $r_i(t) = r_i$. Let $t_{min} = min\{a_1, ..., a_N\}$ and $t_{max} = max\{e_1, ..., e_N\}$. Denote a clustering $\mathrm{C} = \{\mathbf{s}(t_{min}), ..., \mathbf{s}(t_{max})\}$, where $\mathbf{s}(t) = \{s_1(t), s_2(t), ...\}$ is a set of service levels at time t, and $||\mathbf{s}(t)|| \leq L, \forall t \in [t_{min}, t_{max}]$. Then the problem is to find a clustering C^* to

$$Minimize \quad \sum_{i=1}^{N} \sum_{t=a_i}^{e_i} \underset{r_i(t) \to s_j(t)}{c_i(t)} , \tag{3.2}$$

where the second summation is the total cost of clustering request $r_i(t)$ during its lifetime.

Non-Migratory Clustering Problem (NMCP).

Given a set of requests $\mathbf{r}(t) = \{r_1(t), r_2(t), ..., r_N(t)\}$. For any $i \in [1, N]$ and $t \in [a_i, e_i]$, $r_i(t) = r_i$. Let $t_{min} = \min\{a_1, ..., a_N\}$ and $t_{max} = \max\{e_1, ..., e_N\}$. Denote a clustering $\mathrm{C} = \{\mathbf{s}(t_{min}), ..., \mathbf{s}(t_{max})\}$, where $\mathbf{s}(t) = \{s_1(t), s_2(t), ...\}$ is a set of service levels at time t, and $||\mathbf{s}(t)|| \leq L, \forall t \in [t_{min}, t_{max}]$. Since $r_i(t)$ is mapped to a same $s_j(t)$ during its lifetime $t \in [a_i, e_i]$, the problem is to find a clustering C^* to

$$Minimize \quad \sum_{i=1}^{N} (e_i - a_i) \times \underset{r_i(t) \to s_j(t)}{c_i(a_i)} . \tag{3.3}$$

3.3 Optimal Algorithm for Migratory Clustering

3.3.1 Main idea

At first glance, the problem MCP appears very similar to the k-means clustering problem. However, they are fundamentally different. Firstly, the directed distance between a cluster member and the cluster center can be either positive or negative in the k-means problem. In our problem, a service level cannot be lower than the requests assigned to it, otherwise an infinite cost would incur. Secondly, k-means did not consider the lifetime of the requests, thus a current optimal clustering may not be optimal in the future due to the expiration of some members.

Minimize clustering cost at each time instant. Rewrite (3.2) into (3.4), the objective is equivalent to minimize the total cost at each time instant.

$$Minimize \quad \sum_{t=t_{min}}^{t_{max}} \sum_{i=1}^{N} \underset{r_i(t) \to s_j(t)}{c_i(t)} \times I_i(t) \tag{3.4}$$

$$I_i(t) = \begin{cases} 1, & \text{if } t \in [a_i, e_i]; \\ 0, & \text{otherwise.} \end{cases} \tag{3.5}$$

The indicator $I_i(t)$ is introduced since a request is not necessarily alive during the whole period of $[t_{min}, t_{max}]$. A request is only assigned to a service level if it is still alive.

In the following, we further discuss some observations on how to minimize the total cost at each time t. In particular, we first prove that we can narrow down the search space into a finite set without performance loss, and then we present a dynamic programming algorithm to find the optimal clustering.

Pruning search space. In theory, the service levels can be any real number in the range $[0, 1]$, making the problem intractable. Fortunately, we can leverage properties of the problem to shrink the search space into a finite set.

Lemma 1. For MCP, at any time t, the reliability service levels must be selected from the set of requests in order to minimize the cost.

Proof. We prove by contradiction. Without loss of generality, let $\widetilde{\mathbf{r}}(t) = \{r_1(t), r_2(t), ..., r_M(t)\}$ be a set of live requests at time t sorted in non-decreasing order of reliability, and $\mathbf{s}(t) = \{s_1(t)^*, s_2(t)^*, ..., s_K(t)^*\}$ an optimal clustering for $\widetilde{\mathbf{r}}(t)$. Assume there is some $s_j(t)^* \notin \widetilde{\mathbf{r}}(t)$.

Case 1. $s_j(t)^* < r_1(t)$. In this case, if any request is assigned to $s_j(t)^*$, it would introduce a cost of $+\infty$ according to (3.1), which obviously is not optimal; if no request is assigned to $s_j(t)^*$, this service level would be wasted, meaning that we could have a better clustering by choosing another service level, thus $s_j(t)^*$ is not optimal either.

Case 2. $s_j(t)^* > r_M(t)$. In this case, if we move $s_j(t)^*$ to $r_M(t)$, the cost would be further reduced, meaning $s_j(t)^*$ is not optimal.

Case 3. $r_m(t) < s_j(t)^* < r_n(t)$ for some requests $r_m(t)$ and $r_n(t)$. In this case, similar to Case 2, we could reduce the cost by moving $s_j(t)^*$ to $r_m(t)$. By doing this, the costs for clustering requests lower than $r_m(t)$ would reduce, while the costs for clustering requests

higher than $r_n(t)$ remain unchanged. Overall, the total cost would reduce, meaning $s_j(t)^*$ is not optimal. □

Lemma 2. At any time t, the highest-reliability request from the current live set must be selected as a service level.

Proof. This property is straightforward noting that assigning a service lower that than the request leads to infinite cost. If we do not include a service level that equal to the highest live request at current time, there is no service level that can host this request. □

Dynamic programming. Inspired by an edge detection algorithm [29], we adopt a dynamic programming subroutine MCP-Slot to find an optimal clustering for the live requests at time t.

3.3.2 The algorithms

Given a set of live requests at a time slot, Algorithm 2 optimally clusters these requests into different service levels. In the algorithm, $OPT(n, l)$ denotes the optimal cost for clustering the first n requests using l service levels, as shown in Figure 3.2. Line 3 of the algorithm shows that the cost is 0 if there is no request alive at current time. Line 5 means the clustering cost is 0 if there is no service level available thus no clustering is performed. Line 6 denotes that only one service level is available for clustering. In this case, the highest request, i.e., r_n, is selected as the service level according to Lemma 2. Lines 7 to 12 compute the optimal cost for $OPT(n, l)$ for other n and l. If $n \leq l$, the cost is 0 since each request can make itself as a service level. Otherwise, dynamic programming is applied. The key idea is, to cluster n requests into l levels, we first find the optimal cost of clustering the first k requests into $l-1$ levels, then cluster the remaining requests, i.e., $\{r_{k+1}, r_{k+2}, ..., r_n\}$, into the l-th level. According to Lemma 2, the l-th level must be r_n. Among all possible choices of k, we find the one leading to the least cost. This process iterates until N requests are clustered. At last, return the optimal cost of clustering N requests into L levels.

Note that our solution differs from the aforementioned algorithm [29] in several ways. Firstly, rather than restricting the search space into a finite set without justification, we first weaken this assumption and then prune the search space without any performance loss (i.e., Lemma 1). Secondly, the signal processing algorithm uses a cluster-wise fitness function without the notion of cluster center, while we use a request-wise cost function and require that the value of a cluster center is no less than all of its members. Thirdly, the existing algorithm does not specify how to determine $opt(j)$ for $j = 0, 1, ..., n$.

Algorithm 2 MCP-Slot

Sort current live requests in non-decreasing order of the required reliability;
for $l = 0$ to L
    $OPT(0, l) = 0$;
end for
for $n = 1$ to N
    $OPT(n, 0) = 0$;
    $OPT(n, 1) = \sum_{i=1}^{n} |r_n - r_i|$;
end for
for $l = 2$ to L
    for $n = 1$ to N
        if $n \leq l$
            $OPT(n, l) = 0$;
        else
            $OPT(n, l) = \min_{l-1 \leq k \leq n-1} \{OPT(k, l-1) + \sum_{i=k+1}^{n} |r_n - r_i|\}$;
        end if
    end for
end for
return $OPT(N, L)$

Then, we can apply Algorithm 2 to generate the clustering for each time instant between t_{min} and t_{max}. We denote this algorithm as Algorithm 3.

3.3.3 Correctness and complexity of the algorithms

Given that the Algorithm MCP-Slot is a basic element of Algorithm MCP, we analyze the correctness of MCP-Slot.

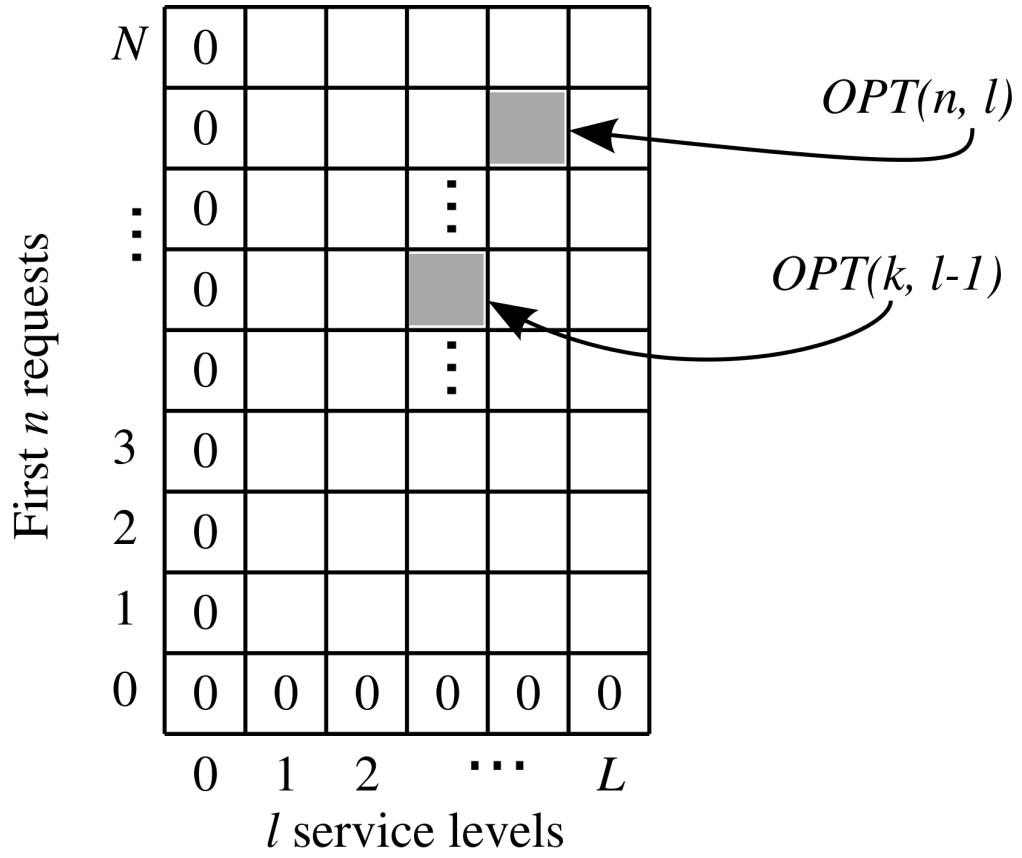


Figure 3.2: The table of solutions for $OPT(n, l)$ in Algorithm 2. Fill the table from bottom to top, left to right. When filling in box, we only need to look at boxes we have already filled in.

Algorithm 3 MCP-Opt

1: t_{min} =min$\{a_i\}$, t_{max} =max$\{e_i\}$;
2: **for** $t = t_{min}$ to t_{max}
3: Current live requests set $\widetilde{\mathbf{r}}(t) = \emptyset$;
4: **for** $i = 1$ to N
5: **if** $a_i \leq t \leq e_i$
6: $\widetilde{\mathbf{r}}(t) = r_i \cup \widetilde{\mathbf{r}}(t)$;
7: **end if**
8: **end for**
9: Call Algorithm 2 to compute a clustering for $\widetilde{\mathbf{r}}(t)$;
10: **end for**

Theorem 3.3.1. *Given a set of live requests at time t, Algorithm 2 gives an optimal clustering for that time instant.*

Proof. For $l = 1$, $OPT(n, 1) = \sum_{i=1}^{n} |r_n - r_i|$ is the only way of clustering due to Lemma 2. For $l > 1$, we just need to show that $OPT(n, l) = \min_{l-1 \leq k \leq n-1} \{OPT(k, l-1) + \sum_{i=k+1}^{n} |r_n - r_i|\}$ is actually optimal. To compute $OPT(n, l)$, we only have two choices for service level l:

Case 1. Leave l: This means that the introduction of the l-th service level will not reduce the cost obtained by clustering n requests over $l-1$ levels. Thus, $OPT(n, l) = OPT(n, l-1)$.

Case 2. Take l: Then we have one more service level to use. The best we can do is to assign k requests into $l-1$ levels, and $n-k$ requests into the l-th level. In determining what k is, we exhaustively search a k such that the cost is minimized. In this case, therefore, $OPT(n, l) = \min_{l-1 \leq k \leq n-1} \{OPT(k, l-1) + \sum_{i=k+1}^{n} |r_n - r_i|\}$. Note that the term $\sum_{i=k+1}^{n} |r_n - r_i|$ is due to Lemma 2.

Taking two cases together, we have $OPT(n, l) = min\{\min_{l-1 \leq k \leq n-1} \{OPT(k, l-1) + \sum_{i=k+1}^{n} |r_n - r_i|\}, OPT(n, l-1)\}$. Due to the cost function we use, it is clear that the clustering cost decreases as the number of service levels increases, thus we can ignore Case 1. □

The complexity of Algorithm 2 is dominated by the loop block from line 7 to line 12. Clearly, it is $O(LN^2)$. Therefore, the complexity of Algorithm 3 is $O(LN^2T)$, where $T = t_{max} - t_{min}$.

3.4 Optimal Algorithm for Non-migratory Clustering

Unlike MCP, which finds an optimal clustering at each time instant independently, NMCP must determine the service level for each request upon its arrival, and each request stays at the same level during its lifetime.

3.4.1 The algorithm

We propose Algorithm 4 to solve the NMCP problem. In the algorithm, $OPT(n, l)$ denotes the optimal cost for clustering the first n requests using l service levels for any possible combinations of n and l. The dynamic programming procedure is similar to Algorithm 2. The major difference lies in the way of mapping a set of requests into a service level. The function MAPPING() maps a set of input requests using one service level. Note that there may be temporal reuse of this service level. Figure 3.3 illustrates this. In the example, there are 7 requests and only one service level is maintained. The function MAPPING() first partitions the requests into 2 disjoint subsets such that any request from first subset will not overlap with any request from second subset in time. These two subsets therefore can use their own service levels without violating the constraint of only one service level maintained at any time. Within each subset, the highest request is selected as the service level. In practice, the number of disjoint subsets determines the degree of temporal reuse of a service level.

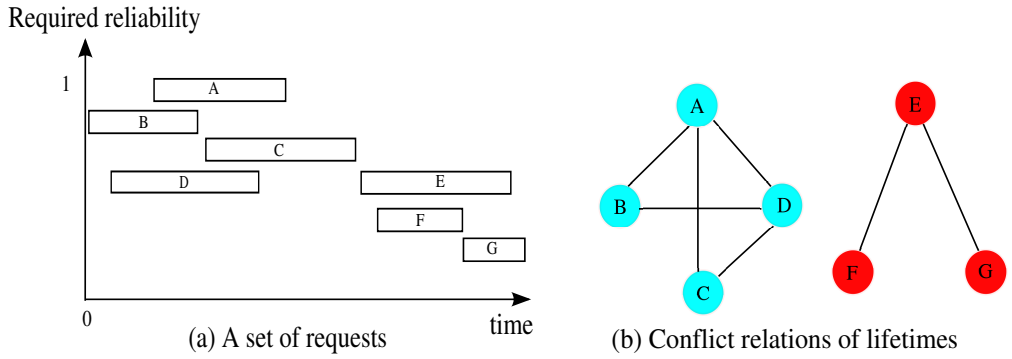


Figure 3.3: An illustration of temporal reuse of a service level.

Algorithm 4 NMCP-Opt

1: **for** $l = 1$ to L
2: $OPT(0, l) = 0$;
3: **end for**
4: **for** $n = 1$ to N
5: $OPT(n, 0) = 0$;
6: $OPT(n, 1)$ =MAPPING($[r_1, ..., r_n]$);
7: **end for**
8: **for** $l = 2$ to L
9: **for** $n = 1$ to N
10: **if** $n \leq l$
11: $OPT(n, l) = 0$;
12: **else**
13: $OPT(n, l) = \min_{l-1 \leq k \leq n-1} \{OPT(k, l-1) +$ MAPPING($[r_{k+1}, ..., r_N]$)$\}$;
14: **end if**
15: **end for**
16: **end for**
17: **return** $OPT(N, L)$
18: MAPPING(**r**)
19: Partition set **r** into disjoint subsets such that requests from different subsets do not overlap with each other in lifetime;
20: **for** each disjoint subset
21: Make the highest request in the subset as the center;
22: Compute the overall cost of this subset;
23: **end for**
24: **return** Total cost of all disjoint subsets;

3.4.2 Correctness and complexity of the algorithm

Theorem 3.4.1. *Given a set of requests, Algorithm 4 gives an optimal clustering for NMCP.*

Proof. For $l = 1$, $OPT(n, 1) =$ MAPPING($[r_1, ..., r_n]$) is the only way of clustering to minimize cost without violating reliability requirement. For $l > 1$, assume we have computed and stored the optimal clustering for the first $n - 1$ requests. To compute $OPT(n, l)$, we assign k requests into $l-1$ levels, and $n-k$ requests into the l-th level. In determining what k is, we exhaustively search a k such that the cost is minimized. We consider the following two cases:

Case 1. There is no overlapping in lifetime between any two of the requests in $[r_{k+1}, ..., r_n]$. In this case, the function MAPPING() partitions the requests into $n - k$ disjoint subsets (i.e., each request make itself a cluster). Thus the cost is 0, which is optimal.

Case 2. There is overlapping in lifetime between some requests in $[r_{k+1}, ..., r_n]$. Without loss of generality, suppose the $n - k$ requests are partitioned into m disjoint subsets. Due to temporal reuse of one service level and exhaustive search of k, the clustering is optimal. □

The function MAPPING() can be easily implemented in $O(N^2)$, thus the complexity of Algorithm 4 is $O(LN^3)$.

3.5 Simulation Results

In this section, we compare the proposed optimal algorithms with a variant of the well known k-means clustering via MATLAB simulations. The k-means clustering does not guarantee reliability by default. To allow it comparable with our algorithms, we modify it by making the service level of each cluster be the highest request in the cluster. We also use M-k-means and N-k-means to denote migratory and non-migratory k-means clustering, respectively. Figure 3.4 shows the total cost of different methods when 100 requests and 100 time slots are simulated. The reliability of each request is randomly generated from $[0, 1]$, and the arrival time and expiration time are randomly picked from $[1, 100]$. The result shows that both MCP-Opt and NMCP-Opt outperform their counterparts M-k-means and N-k-means. The reason is because k-means clustering randomly initializes the cluster centers and does not guarantee global optimality by design. In addition, MCP-Opt outperforms NMCP-Opt since MCP can optimize the clustering on a time slot basis while NMCP does not have such degree of freedom. In all cases, the cost decreases as the number of service levels increases. However, the performance gain becomes negligible when the service level reaches certain threshold. Figure 3.5 shows the total cost of different methods when 20 service levels and 100 time slots are used. As the number of requests increases, the cost also increases in all cases.

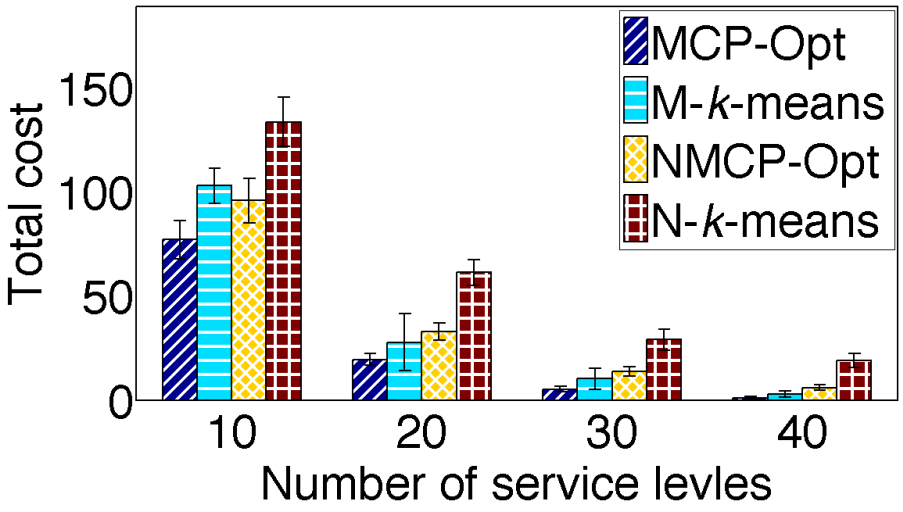


Figure 3.4: Total cost vs. number of service levels L (number of requests $N = 100$).

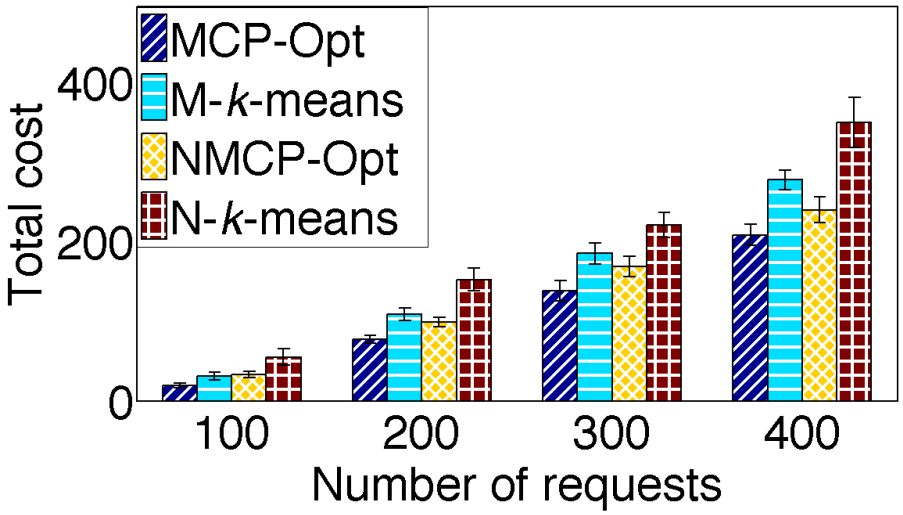


Figure 3.5: Total cost vs. number of requests N (number of service levels $L = 20$).

3.6 Related Work

Clustering has been well studied in machine learning under the context of unsupervised learning, where no labels (or cluster centers) are available beforehand [26]. Among all the clustering algorithms, k-means [27] and its variants [28] are widely used. The basic idea of k-means is to first randomly pick k cluster centers; then assign the input points to the closest centers; update the new cluster centers as the means of the points in each cluster. Repeat this procedure until no further improvement can be made. The k-means algorithm has the following drawbacks. Firstly, it does not guarantee global optimality due to the random initialization of the cluster centers. A large gap between local optimum and global optimum may exist. Secondly, the Euclidean distance between the point and its cluster center is used to measure the clustering quality, which may lead to unsatisfactory clustering.

Clustering has also been considered in signal processing. Edge detection, in particular, finds a set of edge pixels within an image row (or column) as a problem of optimally partitioning that row into a set of segmentations [29] [30]. An algorithm based on dynamic programming is guaranteed to find the global optimum. The proposed algorithm, however, has the following drawbacks. Firstly, it restricts the search space to a finite set assuming that this will not result in significant resolution loss. In practice, the search space is infinite, thus such assumption is open to justification. Secondly, the proposed algorithm does not consider the lifetime of the input. Assuming data points never disappear after arrival, it takes a one-shot clustering for all the input points. In reality, however, old data points may disappear over time and should not be considered for clustering. For example, in real-time scheduling [80], each data packet is associated with a deadline. It is either transmitted before the deadline or discarded if the deadline has been missed.

3.7 Summary

We have propose two algorithms to address the problem of clustering heterogeneous reliability requirements into a limit set of service levels. Our solutions are optimal, and they also provide guaranteed reliability, which is critical for wireless networked control. Future work includes incorporating our algorithms into a running system by considering the wireless channel conditions and medium access control [37].

CHAPTER 4: PROBABILISTIC PER-PACKET REAL-TIME GUARANTEES FOR WIRELESS NETWORKED SENSING AND CONTROL

4.1 Introduction

Embedded wireless networks are increasingly being explored for mission-critical sensing and control in many domains. In real-time augmented vision, for instance, wireless networks can enable the fusion of real-time video streams from spatially distributed cameras to eliminate the line-of-sight constraint of natural human vision and thus enable seeing-through obstacles [40]. In industrial automation, wireless-enabled mobile, pervasive, and reconfigurable instrumentation and the significant cost of planning, installing, and maintaining wired network cables have made wireless networks attractive for industrial monitoring and control; industrial wireless networking standards such as WirelessHART, ISA100.11a, and WIA-PA have also been deployed in practice [41]. Machine-type communication for real-time sensing and control has also become a major focus of the emerging 5G wireless network research and development [47].

Unlike traditional, best-effort wireless networks, reliable and real-time delivery of messages is critical in wireless networked sensing and control (WSC) systems [54]. In industrial WSC systems, for instance, the reliability and timeliness of data delivery directly impact the safety and optimality of sensing and control tasks. Since WSC differs dramatically from the traditional throughput-centric wireless networks due to its mission-critical nature, new design objectives and perspectives are necessary for offering real-time Quality of Service (QoS) in WSC systems.

Per-packet real-time communication guarantee. Wireless communication inherently introduces non-zero packet error probability due to factors such as the uncertainty of lossy links and the contention among different nodes/links, and it introduces non-zero delay due to shared wireless medium and the time taken to transmit each packet. There exist a few recent studies that consider real-time communication guarantees in wireless networks

[38, 39, 50, 51, 70, 71, 72], as shown in Table 4.1. Focusing on the long-term ratio of packets that are delivered before their deadlines, the existing frameworks do not ensure short-term timeliness. For example, consider the scenario shown in Table 4.2. A sensor node periodically sends data to a controller. The controller makes control decision in each period based on the information it collects and requires that at least 90% of the packets are received in time. Otherwise, control failure or even catastrophe may happen. In both cases, the average timely delivery ratios are 90%. In the second case, however, period 2 and period 4 fail to achieve the required delivery ratio due to channel fading, thus the system is subject to transient instability.

In many real-time WSC systems, stringent per-packet real-time communication guarantee is important since it is needed for predictable system performance. For example, in industrial real-time augmented reality applications [55][56][57], real-time delivery of each packet enables seamless, naturalistic 3D reconstruction of real-world scenes (e.g., industrial processes), and consecutive packet loss (which may happen if we only ensure long-term timely delivery ratio) may well lead to uncomfortable human experience. In networked feedback control, consecutive packet loss may well lead to system instability and cause safety concerns; in addition, a packet of sample data will be dropped if the packet has not been successfully delivered by the time a new sample data is collected, and, in this case, the probabilistic guarantee of real-time delivery of each packet enables the modeling of the packet drop process as a random sampling process, thus facilitating the use of random sampling theories to characterize the impact of probabilistic wireless communication on networked control [60][61][62][63][64][65] and facilitating the joint design of wireless networking and networked control as well as the development of field-deployable WSC systems.

Another feature of most WSC systems is that traffic periods (i.e., inter-packet intervals) are allowed to vary within a certain range as long as such changes do not affect critical sensing and control functions [68, 69]. So there exist the challenge and opportunity

of selecting the optimal traffic periods for optimal sensing and control performance while ensuring that the required real-time communication requirements are satisfied.

Contributions. For ensuring probabilistic real-time communication guarantee of each packet as required by real-time WSC systems, this work proposes a probabilistic real-time wireless communication framework that effectively integrates real-time scheduling theory with wireless communication. Major contributions are as follows:

- The per-packet probabilistic real-time QoS is formally modeled. To the best of our knowledge, this paper is the first to propose such notion of timeliness. Existing formulations and designs do not ensure the stringent per-packet assurance as required by WSC systems, since improving the average performance of the network in the long-term does not guarantee short-term QoS.
- R3 mapping is proposed to translate the upper-layer requirement and the lower-layer link reliability into the optimal number of transmission opportunities reserved for each packet.
- An optimal real-time scheduling algorithm and a sufficient and necessary admission test are proposed and formally proved.
- Algorithms are proposed to select packet generation periods to maximize system utility.

Organization of the rest of paper. Section 5.2 summarizes related work. In Section 5.3, we present the system model and problem statement. In Section 4.4, a probabilistic per-packet real-time communication framework is proposed and an optimization problem is formulated and solved. Experimental results are presented in Section 5.5. Section 4.6 addresses the impact of the timescales of real-time communication, multi-channel settings, control signaling unreliability, and distributed implementations. Finally, Section 5.6 concludes the paper.

4.2 Related Work

4.2.1 Real-time scheduling

In traditional real-time systems such as computer processors, the most commonly used scheduling policies are priority-driven, including fixed-priority and dynamic-priority schemes. A fixed-priority algorithm assigns the fixed/same priority to all jobs (i.e., instances of a task) in each task. For example, the Rate Monotonic algorithm [44] assigns priority based on the rate (or period): the shorter the period of a task, the higher its priority. Another example is the Deadline Monotonic algorithm [45], which assigns a higher priority to a job with a shorter relative deadline. In contrast, dynamic-priority scheduling algorithms assign priorities dynamically. The Earliest Deadline First (EDF) algorithm [44] is a preemptive priority-based dynamic scheduling algorithm. The scheduler always schedules the active task with the earliest absolute deadline.

Time-Sensitive-Networking (TSN) [41] has recently been proposed to enable real-time communication in industrial networks. Focusing on extending the IEEE 802.1-standards-based Ethernet networks, TSN-oriented work examines issues in real-time wired communication instead of real-time wireless communication.

Real-time scheduling for wireless network has drawn increasing interest recently. By leveraging real-time scheduling theory for processors and incorporating unique wireless characteristics, a Conflict-aware Least Laxity First algorithm was proposed for WirelessHART [50]. Analysis for EDF in wireless sensor-actuator networks was investigated [51]. The key insight from these work is to draw an analogy between wireless channel and a processor such that the classic real-time theories can be leveraged. However, they did not reveal the inherent relationship between the application requirement, link reliability, and resource reservation. They also did not ensure short-term probabilistic real-time packet delivery. There also exist other research efforts that consider real-time communication guarantees in wireless networks [38][39]. Modeling the delivery ratio of packets that meet their deadlines as a long-term, asymptotic average over multiple (and infinitely many) transmission periods, the proposed

frameworks in those studies do not assure short-term timeliness either. IEEE 802.15.4 or 802.11 based real-time protocols have also been investigated for mission-critical industrial systems. A real-time message-scheduling algorithm was proposed for IEEE 802.15.4-based industrial WSNs [70]. An analysis of the real-time performance that can be achieved in QoS-enabled 802.11 networks has been carried out [71]. RT-WiFi is a TDMA data link layer protocol based on 802.11 physical layer to provide deterministic timing guarantee on packet delivery in wireless control systems [72]. Again, these methods have not been designed for applications that need per-packet real-time guarantees. Probabilistic real-time scheduling methods have also been proposed for WirelessHART [34] and fault-tolerant EDF scheduling [35], but they have not considered the real-time wireless scheduling problems studied in this paper either.

4.2.2 Reliability-oriented wireless transmission scheduling

Providing predictable link reliability is critical in WSC systems. To guarantee that packets are delivered at requested reliability levels, Zhang et al. proposed PRKS for wireless networked control [37]. PRKS is a TDMA-based distributed protocol to enable predictable link reliability based on the Physical-Ratio-K (PRK) interference model [36]. Through a control-theoretic approach, PRKS instantiates the PRK model parameters according to in-situ network and environment conditions so that each link meets its reliability requirement. In particular, PRKS defines a conflict graph for a given wireless network: a node in the graph denotes a link with data transfer in the network, and a link exists between two nodes in the graph if the corresponding links in the network interfere with each other according to the PRK model. Under the condition that link reliability is ensured, PRKS schedules as many nodes as possible in the conflict graph. Based on the predictable link reliability as ensured by algorithms such as PRKS, this paper develops the framework and algorithm for ensuring predictable per-packet real-time communication.

4.3 System Model and Problem Statement

This section presents the system models and real-time communication scheduling problem considered in this paper.

4.3.1 Communication model

Similar to [52][53], the system consists of n wireless sensors and one controller, i.e., access point (AP), as shown in Figure 4.1. All the sensors are within the communication

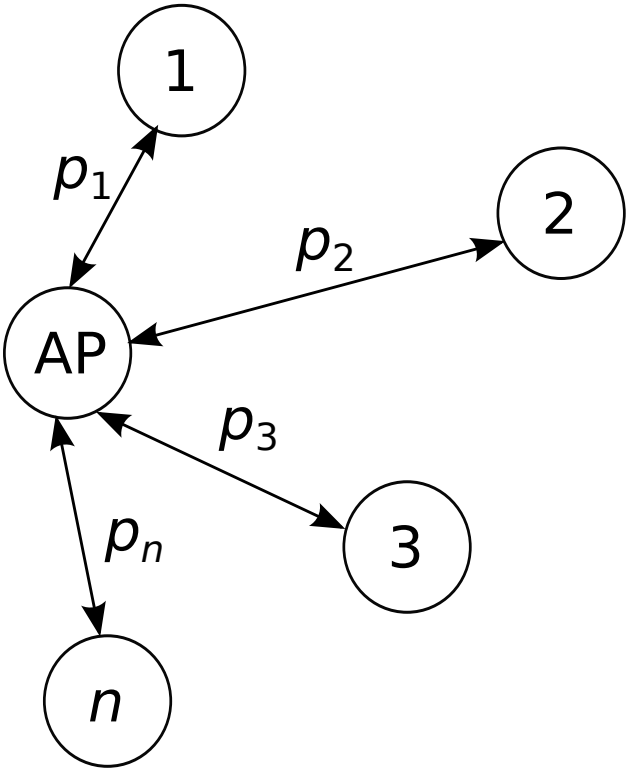


Figure 4.1: System model: one AP and n sensors.

range of the controller. Time is slotted and synchronized across the controller and sensors, and the wireless transmissions between the sensors and controller are scheduled in a TDMA manner. For simplicity of presentation, our discussion focuses on a centralized master-slave architecture where the scheduler lies in the AP. At the beginning of each time slot, the AP broadcasts a short control signaling message to indicate the sensor that shall transmit in the current time slot. For simplicity of presentation, we assume that the control signaling is

reliable (e.g., through transmission power control). (We will discuss how to address potential unreliability in control signaling in Section 4.6.3, and we will discuss potential distributed approaches in Section 4.6.4.) The duration of a time slot is the time required for the controller to send the control packet plus the time for a sensor to transmit a data packet of certain fixed length. All the time slots are of the same length. To avoid transmission collision, there can be at most one active sensor transmitting at each time slot.

The link reliability between a sensor i and the controller is p_i, meaning that a packet transmission succeeds in probability p_i. Similar to [46], the event of successful transmissions along a link is assumed to be i.i.d. over time, thus whether a packet can be successfully delivered at a time slot is independent of the status of earlier transmissions. Note that the communication along a wireless link may well be i.i.d. in practice when mechanisms such as transmission power control is used to ensure a certain communication reliability at each time instant based on in-situ wireless channel conditions. In addition, as we will discuss in more detail in Section 5.4.2, our methodologies of probabilistic real-time scheduling also applies to other wireless link models such as Markovian models and links with deep fading.

4.3.2 Probabilistic QoS model

The traffic on link i is characterized by a 4-tuple $(T_i, T_{i,max}, D_i, P_i)$:

- Period T_i: sensor i generates one data packet every T_i time slots. Wireless networked control can usually tolerate a flexible traffic period, provided the period is chosen below an upper bound $T_{i,max}$, since a traffic period too large may not generate sufficient packets for the control application. In a very general framework, T_i is the design variable to represent the possible choices of the designer at the stage of the system design.
- Relative deadline D_i: for client i, each packet is associated with a relative deadline D_i in units of time slots. A packet arriving at time slot t should be successfully delivered no later than time slot $t + D_i$; otherwise, the packet is dropped. Since the node always sends new sensing and control signal, $D_i \leq T_i$. In this paper, D_i

and T_i are assumed to be equal, and this scenario is usually referred to as *implicit deadline* in classic real-time systems and finds its application in various areas [48]. Note that heterogeneous deadlines across different links are considered since more stringent deadlines are assigned to more critical sensor nodes in practice.

- Application requirement P_i: due to inherent dynamics and uncertainties in wireless communication, real-time communication guarantees are probabilistic in nature. Thus a probabilistic QoS metric is defined.

  Definition 1 Link i meets probabilistic per-packet real-time guarantee if $\forall j$, $Prob\{F_{ij} \leq D_i\} \geq P_i$, where F_{ij} is the delay (measured in the number of time slots) in successfully delivering the j-th packet of link i. Intuitively, the probabilistic per-packet real-time guarantee for link i ensures that in an *arbitrarily* long execution history, a *randomly* selected packet of link i will meet the deadline D_i at least in probability P_i.

Remarks. Considering TDMA scheduling in this work, the parameters T_i, $T_{i,max}$, and D_i are all integers for each link i. The proposed probabilistic QoS model differs fundamentally from the classic soft-real-time model where the QoS metric is the long-term timely delivery ratio (i.e., the long-term ratio of the number of packets successfully delivered in time to the total number of packets transmitted).

4.3.3 Problem statement

Based on the system model, an important question to ask is as follows. Given a set of n links with each link i having a link reliability p_i and periodic traffic $(T_i, T_{i,max}, D_i, P_i)$, where $D_i = T_i$ and T_i can vary in a range with an upper bound $T_{i,max}$, are the set of links schedulable to meet the probabilistic per-packet real-time communication requirements? If yes, find a set of optimal periods such that the system utility (to be defined precisely in Section 4.4.5) is maximized while ensuring real-time schedulability; If no, indicate the infeasibility.

To answer this question, a probabilistic real-time wireless communication framework is proposed in the next section.

4.4 Wireless Scheduling Framework for Probabilistic Per-Packet Real-Time Communication Guarantee

4.4.1 Overview

Compared with scenarios where every packet transmission is successful, probabilistic communication errors in wireless systems lead to increased delays in delivering packets to their destinations and thus introduce additional deadline violations and packet drops. This additional packet drop may well cause non-negligible degradation in the quality of data delivery. Thus, the decision regarding whether a set of links is schedulable must take into consideration the inherent nature of probabilistic wireless communication.

For example, consider a simple case where there are two links and each link generates one packet in a period of 3 time slots. If there are no channel errors (as assumed in classic real-time systems), the required delay guarantee can be provided to every packet of every link (Figure 4.2(a)). In the presence of slight channel errors (Figure 4.2(b)), the delay experienced by each packet is increased by at most one time slot, but both links can still deliver their packets on time. Figure 4.2(c) shows the case when the links are not schedulable. At the third time slot, both links need to send out the packet since this is the last slot in current period. However, only one can send, and the other has to drop its packet. Thus packet drop happens.

For a packet that needs to be successfully delivered across a link i within deadline D_i and in probability no less than P_i, the requirement can be decomposed into two sub-requirements: 1) successfully delivering the packet in probability no less than P_i, and 2) the time taken to successfully deliver the packet is no more than D_i if it is successfully delivered. Given a specific link reliability p_i, the first sub-requirement translates into the required minimum number of transmission opportunities that need to be provided to the transmission of the packet. Then, the second sub-requirement requires that these minimum number of transmission opportunities are used within deadline D_i. That is, the original problem of

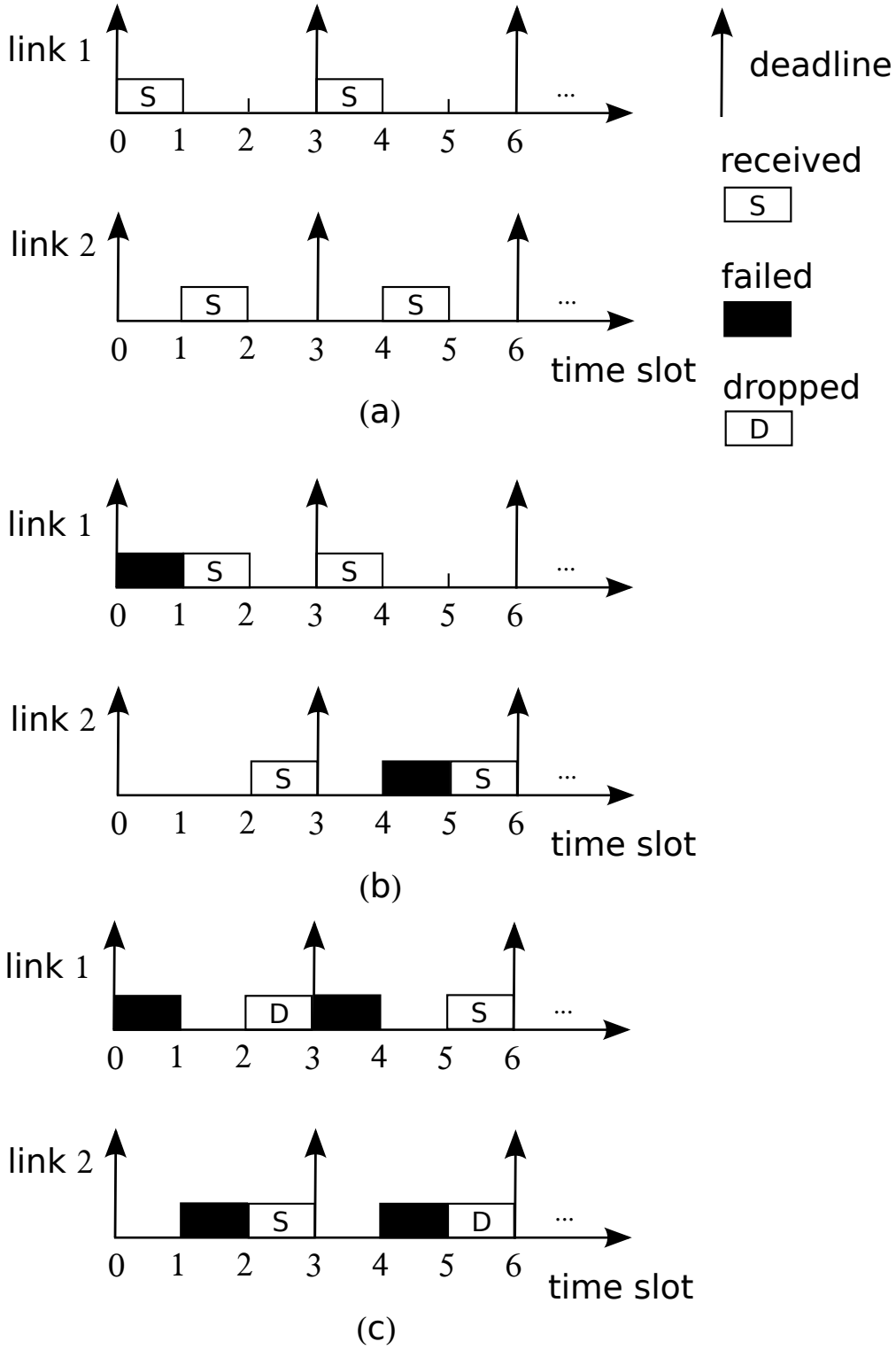


Figure 4.2: An example showing the uncertainty of wireless channel.

ensuring probabilistic per-packet real-time delivery is decomposed into two sub-problems of ensuring probabilistic delivery reliability and real-time packet transmission respectively.

With the above observation, an architecture for solving the probabilistic per-packet real-time scheduling problem of Section 5.3.2 is shown in Figure 4.3. Given a set of links with

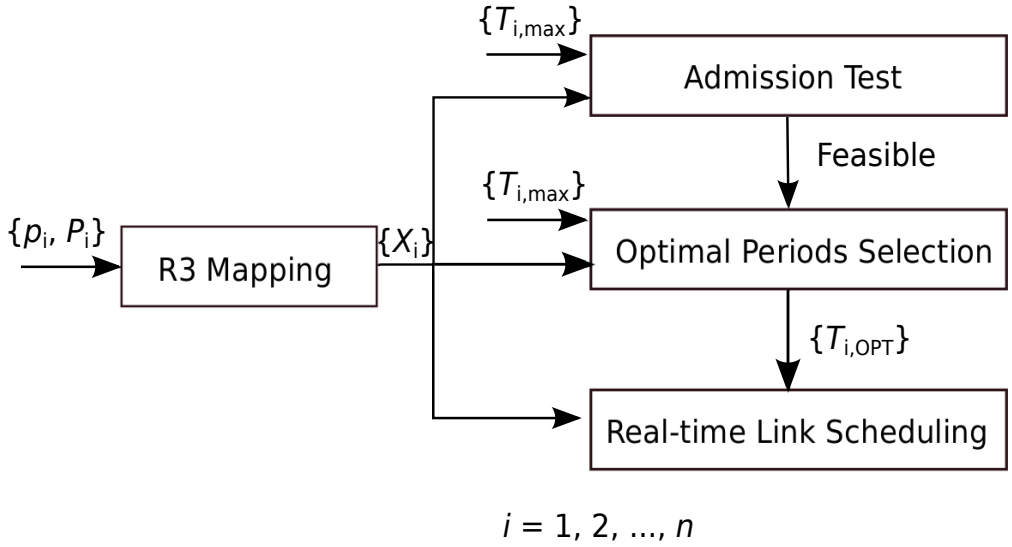


Figure 4.3: Architecture of the proposed framework.

each link i having link reliability p_i, application requirement P_i, and traffic $(T_i, T_{i,max}, D_i)$, the "R3 Mapping" component uses p_i and P_i of link i to compute X_i, i.e., the number of time slots (i.e., transmission opportunities) needed to ensure the successful delivery of a packet along link i in probability P_i. Then the "Admission Test" component determines whether the links are schedulable or not based on X_i and the input $T_{i,max}$. If schedulable, a set of optimal periods $T_{i,OPT}$ are computed based on X_i, T_i, and $T_{i,max}$. Then the links are scheduled using an optimal real-time scheduling algorithm based on the optimal traffic periods, the transmission opportunities reserved, and delivery deadlines. In the following, we elaborate on each component of the framework.

4.4.2 Requirement-reliability-resource (R3) mapping

R3 mapping is invoked to compute the number of transmission opportunities needed to ensure successful delivery a packet along each link with the required probability guarantee.

Theorem 4.4.1. *Given link reliability p_i and application requirement P_i, the minimum number of time slots (or transmission opportunities) needed to ensure successful delivery of a packet for link i in probability P_i, denoted as X_i, is as follows:*

$$X_i = \lceil log_{1-p_i}(1 - P_i) \rceil. \tag{4.1}$$

Proof. Let

$$\begin{aligned} Q_i &= Prob\{\text{A packet delivered within } X_i \text{ transmissions}\} \\ &= 1 - Prob\{\text{Fail all } X_i \text{ transmissions}\} \\ &= 1 - \underbrace{(1-p_i)(1-p_i)...(1-p_i)}_{X_i} \\ &= 1 - (1-p_i)^{X_i}. \end{aligned} \tag{4.2}$$

To achieve the application requirement P_i,

$$Q_i = 1 - (1-p_i)^{X_i} \geq P_i. \tag{4.3}$$

Thus, the minimum number of transmissions needed is $X_i = \lceil log_{1-p_i}(1 - P_i) \rceil$.

□

Note that the X_i reserved time slots do not have to be X_i consecutive time slots, and, for real-time packet delivery, the X_i time slots only have to be before the delivery deadline of the packet. By R3 mapping, the number of transmission opportunities needed to ensure probabilistic packet delivery success (i.e., X_i) is derived from the channel statistics and the application requirement, thus transforming the probabilistic real-time delivery requirement into a problem involving the reservation of a deterministic number of transmission opportunities for each link. This transformation facilitates the application/extension of traditional real-time CPU scheduling theory to the real-time wireless transmission scheduling as we will explain in Sections 4.4.3 and 4.4.4.

The above derivation of X_i is based on the link model where the per-packet transmission success probability along link i is p_i, which is a valid model for cases where the per-packet transmission reliability is controlled at a certain level (e.g., p_i) using techniques such as channel-aware transmission power control. For cases where other models such as Markovian models or empirical models of packet transmission successes [66][67] may be more appropriate (e.g., when no mechanism is adopted to ensure a certain per-packet communication reliability in the presence of channel deep fading), the derivation of X_i needs to be revised accordingly, but the rest of our probabilistic real-time scheduling framework would still apply. As a first step in studying scheduling with probabilistic per-packet real-time guarantees, we focus on scenarios where the per-packet transmission reliability is controlled at p_i, and we relegate the detailed studies of alternative link models as future work.

4.4.3 Scheduling policy

Priority-based scheduling algorithms are usually used in real-time systems [44][45]. Consider the better resource utilization and agility that a dynamic-priority algorithm can achieve, in this paper, a dynamic-priority scheduling algorithm based on the Earliest Deadline First (EDF) concept is used. In the algorithm, we define the deadline of a link as the deadline of the packet that needs to be delivered across the link; if there is no packet that is yet to be delivered across the link at a time slot, we regard the link's deadline as positive infinity at the time slot. Then, at each time slot, the link with the earliest deadline is scheduled by the algorithm. This can be represented by Algorithm 5.

In the following, the proposed scheduling algorithm is shown to be feasibility-optimal.

Theorem 4.4.2. *(Feasibility Optimality of Algorithm 5) Algorithm 5 can produce a feasible schedule for a set of links L with arbitrary deadlines D_i on a single wireless channel if and only if there exists any algorithm that can schedule the transmissions of links L to ensure the required probabilistic per-packet real-time guarantee.*

Proof. The theorem is proved by transforming the wireless packet transmission scheduling problem into a real-time CPU scheduling problem [48] in the following manner: treat each

Algorithm 5 Scheduling algorithm

```
1: for each time slot
2:     target_link_index = 0;
3:     min_deadline = ∞;
4:     for i = 1 to n
5:         if link[i].deadline < min_deadline
6:             min_deadline = link[i].deadline;
7:             target_link_index = i;
8:         end if
9:     end for
10:    Schedule link[target_link_index];
11: end for
```

link i as a task i; then each packet j along link i corresponds to a job j of task i, and the number of transmission opportunities X_i required to ensure the probabilistically successful delivery of a packet corresponds to the execution time C_i in task scheduling. Accordingly, the traffic demand $\{(T_i, T_{i,max}, D_i, P_i)\}_{i=1...n}$ can be transformed into a task system $\mathcal{J} = \{(T_i, D_i, X_i)\}_{i=1...n}$ of n periodic tasks, with each task i specified by its period T_i, relative deadline D_i, and execution time $X_i = \lceil log_{1-p_i}(1 - P_i) \rceil$. In a task i $(i = 1 \dots n)$, a job is repeated every T_i time slots, and every instance of the repeating job has a relative deadline of D_i and an execution time of X_i. Then, Algorithm 5 is the same as the EDF algorithm for preemptive CPU scheduling except for the following differences: 1) the task system $\mathcal{J}$ is such that the units of time is a time slot, and the period of a task as well as the arrival time, relative deadline, and execution time of each job of the task are all integers (instead of being real numbers as in a preemptive CPU scheduling problem setup); 2) given that the transmission of a packet in a time slot cannot be interrupted in general, the transmission along a link i can only preempt the transmission along another link i' at the boundary of two consecutive time slots, thus a task in the corresponding task system can be preempted only at the boundary of consecutive time slots (instead of being at any arbitrary point in time as in a preemptive CPU scheduling problem setup), and the task system is a limited preemptive system. Just as the EDF scheduling algorithm is optimal for preemptive single CPU scheduling, Algorithm

5 is optimal for the aforementioned task system $\mathcal{J} = \{(T_i, D_i, X_i)\}_{i=1...n}$, and the proof is based on induction as follows.

Let Δ denote an arbitrarily small positive number of time slots. Consider a schedule S for $\mathcal{J}$ in which all deadlines are met, and let $[t_0, t_0 + \Delta)$ denote the first time interval over which this schedule makes a scheduling decision different from the one made by Algorithm 5. Suppose that a job/packet $j_1 = (a_1, e_1, d_1)$ with arrival time a_1, execution time e_1, and deadline d_1 is scheduled in S over this interval, while another job $j_2 = (a_2, e_2, d_2)$ is scheduled in Algorithm 5. Since S meets all the deadlines, it is the case that S schedules j_2 to completion by the end of time slot d_2. In particular, this implies that S schedules j_2 for an amount at least Δ prior to d_2. But by the definition of Algorithm 5, $d_2 \leq d_1$; hence, S schedules j_2 for an amount at least Δ prior to d_1 as well. Now the new schedule S' obtained from S by swapping the executions of j_1 and j_2 of length Δ time slots each would agree with Algorithm 5 over $[0, t_0 + \Delta)$. The proof of optimality of Algorithm 5 now follows by induction on time, with S' playing the role of S in the above argument. □

4.4.4 Admission policy

Given an arbitrary set of links, it is not always possible to find a schedule to meet the probabilistic deadlines. Therefore, an admission policy is needed to determine whether the set of links is feasible. In this paper, a reservation-based admission policy is used. Since link i generates one packet every T_i time slots, by R3 mapping, it is known that it needs to reserve X_i time slots to ensure the probabilistic per-packet real-time delivery. Thus the admission test is such that if the test is passed, each link is guaranteed for a certain number of reserved transmission opportunities.

Definition 2 (Work density) The work density of link i, denoted by ρ_i, is defined as the ratio of the number of transmission opportunities needed to ensure the required probabilistic real-time communication guarantee along link i to the traffic period of link i. That is,

$$\rho_i = \frac{X_i}{T_i}. \tag{4.4}$$

Then, we have

Theorem 4.4.3. *A set of n links is schedulable if and only if*

$$\sum_{i=1}^{n} \rho_i \leq 1. \tag{4.5}$$

Proof. As we have shown in the proof of Theorem 2 earlier, the real-time wireless transmission scheduling problem considered in this work can be transformed into a special preemptive CPU scheduling problem, and the EDF-based Algorithm 5 is optimal just as the EDF algorithm is optimal for real-time preemptive single CPU scheduling. For real-time preemptive single CPU scheduling, Liu [48] has shown, by Theorem 6.1 of [48], that a task system is schedulable if and only if the total work density of the task system is no more than 1. The proof of Theorem 6.1 on pages 124-126 of Reference [48] can be directly applied to prove Theorem 3 here, since the proof of Theorem 6.1 in Reference [48] applies to the special task system $\mathcal{J}$ presented in the proof of Theorem 2 earlier. For conciseness of presentation, we skip the detailed proof here.

□

Since the upper bound of period for each link is known, besides Theorem 3, a simple infeasibility test can be applied to avoid further searching by the following lemma.

Lemma 1 Given a set of n links, if the schedulability condition from Theorem 3 is not satisfied for $T_{i,max}$, $i = 1, .., n$, then it is not satisfied for any $T_i < T_{i,max}$.

Proof. If the link set fails the schedulability test in Theorem 3, then

$$\sum_{i=1}^{n} \frac{X_i}{T_{i,max}} > 1.$$

Using any $T_i < T_{i,max}$ will lead to

$$\sum_{i=1}^{n} \frac{X_i}{T_i} > \sum_{i=1}^{n} \frac{X_i}{T_{i,max}} > 1.$$

This concludes the proof. □

Then the admission test is summarized as Algorithm 6.

Algorithm 6 Admission test

1: $total\_density \leftarrow 0$;
2: **for** $i = 1$ to n
3: Compute $X_i = \lceil log_{1-p_i}(1 - P_i) \rceil$;
4: $total\_density \leftarrow total\_density + \frac{X_i}{T_{i,max}}$;
5: **if** $total\_density > 1$
6: **return** Infeasible;
7: **end if**
8: **end for**
9: **return** Feasible;

4.4.5 Selection of packet generation period

In this section, we address the issue of selecting the optimal packet generation periods while ensuring schedulability for a set of feasible links. Without loss of generality, assume the utility function is concave and monotonically increasing. Since T_i is an integer (i.e., number of time slots), the optimization problem to solve becomes a discrete optimization problem. More precisely, this problem is:

$$\underset{T_i, i=1...n}{\text{Maximize}} \quad \sum_{i=1}^{n} U_i(\frac{1}{T_i}), \tag{4.6}$$

subject to:

$$\sum_{i=1}^{n} \frac{X_i}{T_i} \leq 1, \tag{4.7}$$

$$T_i \leq T_{i,max}, \quad i = 1, ..., n. \tag{4.8}$$

The maximization is with respect to T_i's. The utility function U_i increases as the period T_i decreases, which is quite intuitive. The optimal solution for this problem can be obtained using Branch-and-Bound methods, especially when the problem size is small. Given that the time complexity of branch-and-bound methods is as high as that of exhaustive search in the worst case (i.e., $O({T_{i,max}}^n)$), however, here we also turn to pursue a more efficient

approximation algorithm for problems of larger sizes. The basic idea is to transform the problem by replacing the discrete variable T_i with real number t_i, then solve the new problem using convex optimization technique. We first transform the problem as

$$\underset{t_i, i=1...n}{\text{Maximize}} \quad \sum_{i=1}^{n} U_i(\frac{1}{t_i}), \tag{4.9}$$

subject to:

$$\sum_{i=1}^{n} \frac{X_i}{t_i} \leq 1, \tag{4.10}$$

$$t_i \leq T_{i,max}, \quad i = 1, ..., n. \tag{4.11}$$

For ease of mathematical derivation, replace $\frac{1}{t_i}$ above with f_i, where f_i is the sensing frequency. Then the optimization problem can be rewritten as:

$$\underset{f_i, i=1...n}{\text{Maximize}} \quad \sum_{i=1}^{n} U_i(f_i), \tag{4.12}$$

subject to:

$$\sum_{i=1}^{n} X_i f_i \leq 1, \tag{4.13}$$

$$f_{i,min} \leq f_i, \quad i = 1, ..., n, \tag{4.14}$$

where $f_{i,min} = \frac{1}{T_{i,max}}$ is the lower bound of the sensing frequency. By transforming E.q. (4.10) into E.q. (4.13), we perform derivation over f_i rather than t_i when apply the KKT conditions.

Introducing Lagrange multipliers λ for the inequality constraints $\sum_{i=1}^{n} X_i f_i \leq 1$, and λ_i for the inequality constraints $f_{i,min} \leq f_i$, it follows that

$$I(f_1, ..., f_n) = \sum_{i=1}^{n} U_i(f_i) - \lambda(\sum_{i=1}^{n}(X_i f_i - 1)) - \sum_{i=1}^{n}(\lambda_i(f_{i,min} - f_i)).$$

Then the KKT conditions can be obtained as

$$\begin{cases} \nabla_f I(f_1, ..., f_n) = 0, \\ \sum_{i=1}^{n} X_i f_i - 1 \leq 0, \\ f_{i,min} - f_i \leq 0, \quad i = 1, ..., n, \\ \lambda(\sum_{i=1}^{n} X_i f_i - 1) = 0, \\ \lambda_i(f_{i,min} - f_i) = 0, \quad i = 1, ..., n, \\ \lambda \geq 0, \\ \lambda_i \geq 0, \quad i = 1, ..., n. \end{cases}$$

The first condition can be further expanded into n equations:

$$\nabla_f I(f_1, ..., f_n) = \begin{bmatrix} \frac{\partial I(f_1, ..., f_n)}{\partial f_1} \\ \frac{\partial I(f_1, ..., f_n)}{\partial f_2} \\ \vdots \\ \frac{\partial I(f_1, ..., f_n)}{\partial f_n} \end{bmatrix} = \begin{bmatrix} \frac{d}{df_1} U_1(f_1) - \lambda X_1 + \lambda_1 \\ \frac{d}{df_2} U_2(f_2) - \lambda X_2 + \lambda_2 \\ \vdots \\ \frac{d}{df_n} U_3(f_n) - \lambda X_n + \lambda_n \end{bmatrix} = 0.$$

For simplicity, rewrite the above equations as

$$\begin{cases} \frac{d}{df_i}U_i(f_i) - \lambda X_i + \lambda_i = 0, \quad i = 1, ..., n, & (4.15a) \\ \sum_{i=1}^{n} X_i f_i - 1 \leq 0, & (4.15b) \\ f_{i,min} - f_i \leq 0, \quad i = 1, ..., n, & (4.15c) \\ \lambda(\sum_{i=1}^{n} X_i f_i - 1) = 0, & (4.15d) \\ \lambda_i(f_{i,min} - f_i) = 0, \quad i = 1, ..., n, & (4.15e) \\ \lambda \geq 0, & (4.15f) \\ \lambda_i \geq 0, \quad i = 1, ..., n. & (4.15g) \end{cases}$$

Rewrite E.q. (4.15a) as $\lambda = \frac{\frac{d}{df_i}U_i(f_i) + \lambda_i}{X_i}$. Since the utility function $U_i(f_i)$ is monotonically increasing with f_i, $\frac{d}{df_i}U(f_i) > 0$. In addition, $X_i > 0$ and $\lambda_i \geq 0$, thus $\lambda > 0$. Then, E.q. (4.15d) implies that $\sum_{i=1}^{n} X_i f_i = 1$. In this case, the problem becomes:

$$\underset{f_i, i=1...n}{\text{Maximize}} \quad \sum_{i=1}^{n} U_i(f_i), \tag{4.16}$$

subject to:

$$\sum_{i=1}^{n} X_i f_i = 1, \tag{4.17}$$

$$f_{i,min} \leq f_i, \quad i = 1, ..., n. \tag{4.18}$$

Thus, the problem to be addressed contains only the equality and lower bound constraints. For many simple utility functions, closed form solutions can be obtained; in general, a set of optimal periods $t_{i,OPT}$ can be obtained in time $O(nlogn)$ by using the OPT-L algo-

rithm [49]. Then we can approximate the optimal period $T_{i,OPT}$ by taking ceiling on $\frac{1}{f_{i,OPT}}$ for $1 \leq i \leq n$, that is, letting $T_{i,OPT} = \lceil \frac{1}{f_{i,OPT}} \rceil$.

Algorithm 7 summarizes our approach to selecting the packet generation periods.

Algorithm 7 Selection of packet generation period

1: **if** problem size is small (e.g., $n \leq 30$)
2: Use branch-and-bound method to solve (4.6)-(5.13).
3: **else**
4: Use the OPT-L algorithm [49] to solve (4.16)-(4.18);
5: $T_{i,OPT} = \lceil \frac{1}{f_{i,OPT}} \rceil, 1 \leq i \leq n$.
6: **end if**

4.5 Simulation Experiment

The proposed admission control and scheduling algorithm are denoted as prob-QoS. In this section, the performance for prob-QoS is evaluated via Matlab simulation. We first discuss the simulation methodology and then show the results.

4.5.1 Simulation methodology

- **Packet loss model**. The packet reception is modeled with Bernoulli trials, i.e., a transmission will succeed with probability p, where $0 \leq p \leq 1$.
- **Performance metrics**. Regarding the performance metrics, *QoS satisfaction ratio* and *idleness ratio* are used. First, QoS satisfaction ratio is the ratio that the required per-packet QoS is guaranteed. It is the primary goal of this work. Second, since the number of time slots needed X_i is estimated based on the channel statistics, it is possible that a slot assigned to a link at certain time is not used since it has already successfully delivered the packet. In this case, the remaining slots reserved for this link will be idle. The idleness ratio is the ratio between the idle slots and the total slots reserved. In the premise of QoS satisfaction, a smaller idleness ratio is favored since it implies higher channel utilization.

- **Application settings**. For reflecting the mission-critical scenarios, consider the cases when the application requirement P_i for each link is set to 90%, 95%, or 99% respectively. To understand the impact of link reliability p_i, experiments with varying p_i are also performed. The number of links n is 16, period T and deadline D are 100 time slots. This setting is to ensure there is feasible solution for the admission test.
- **Baseline algorithm**. To evaluate the per-packet real-time guarantee, the proposed scheme is compared with the state-of-the-art debt-based algorithm [38]. The debt of a client i at time t is defined as the number of time slots needed by i minus the actual number of time slots it has transmitted by t. At each time slot, the AP schedules the client with largest debt.

4.5.2 Simulation results

Figure 4.4 compares the QoS satisfaction ratios of prob-QoS and debt-based algorithm when the link reliability $p = 0.6$. As shown in the figure, prob-QoS can always guarantee per-packet QoS for all requirements, while the debt-based algorithm fails all cases. The reason is because prob-QoS reserves enough resource for each packet by R3 mapping and admission test, while the debt-based algorithm only focuses on long-term average performance and ignores short-term fluctuations

Figure 4.5 evaluates the QoS satisfaction ratio by fixing the application requirement to 99% and varying the link reliability. As shown in the figure, prob-QoS always ensures 99% QoS satisfaction ratio while the debt-based algorithm fails to.

For the idleness ratio, similar experiments are conducted. Figure 4.6 shows the case when the link reliability is 0.6 for both algorithms. As shown in the figure, prob-QoS has a larger idleness ratio than the debt-based algorithm. This is because the primary design objective of prob-QoS is to ensure per-packet real-time QoS. By Definition 1, such per-packet QoS must be satisfied for a *randomly* picked packet in an *arbitrarily* long execution history. Mission-critical applications require predictable behaviors under all possible circumstances, thus the conservativeness of reservation is the cost that needs to be taken, otherwise the

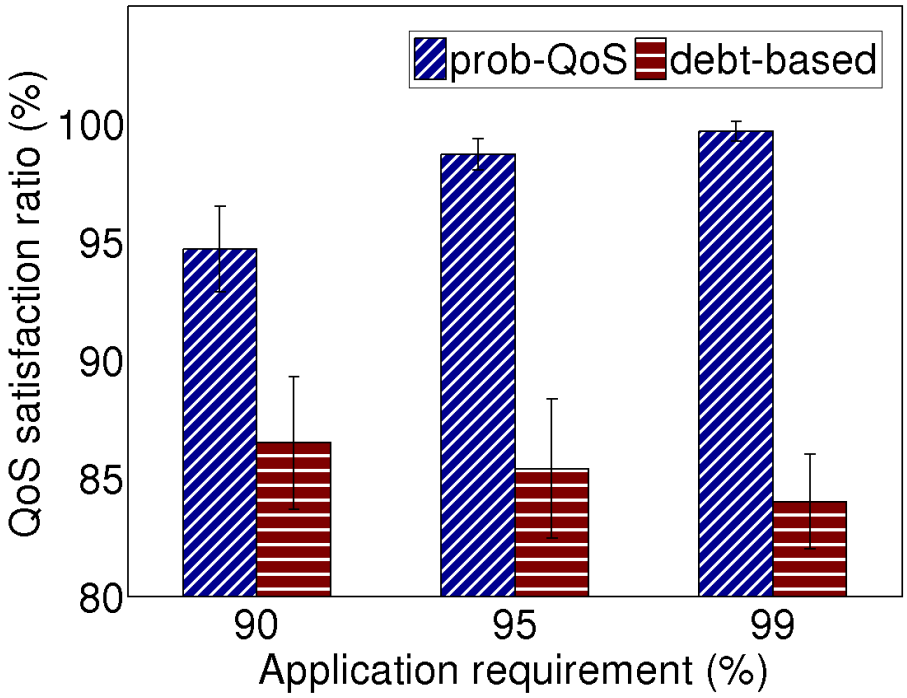


Figure 4.4: QoS satisfaction ratio: prob-QoS vs debt-based algorithm with different application requirements.

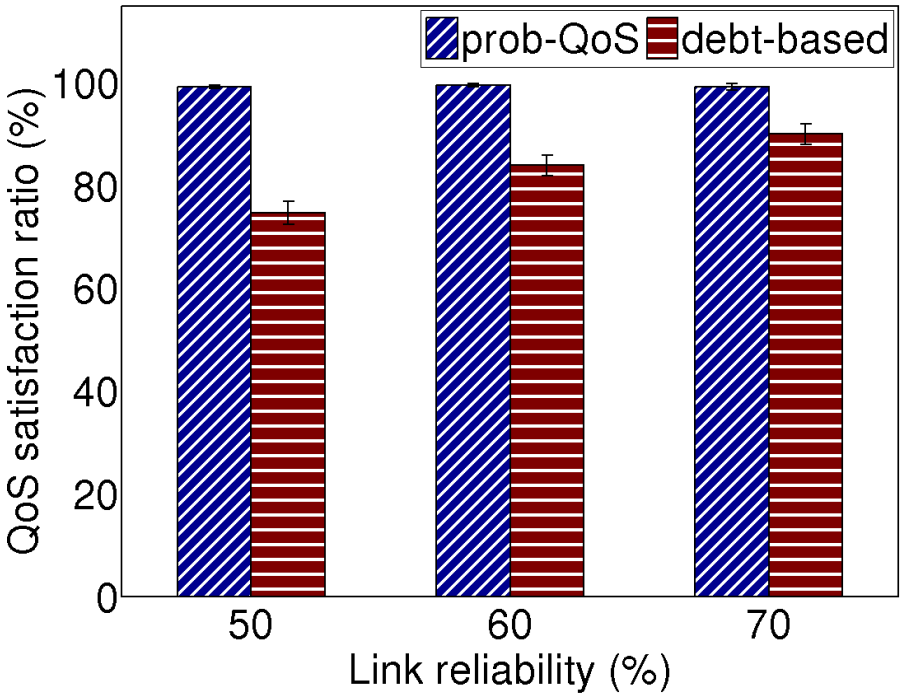


Figure 4.5: QoS satisfaction ratio: prob-QoS vs debt-based algorithm with different link reliability.

required QoS cannot be met. Figure 4.7 shows similar results when the application requirement is set to 99% and link reliability varies. In Section 4.6.1, the inherent tradeoff between throughput and real-time guarantees will be discussed in detail.

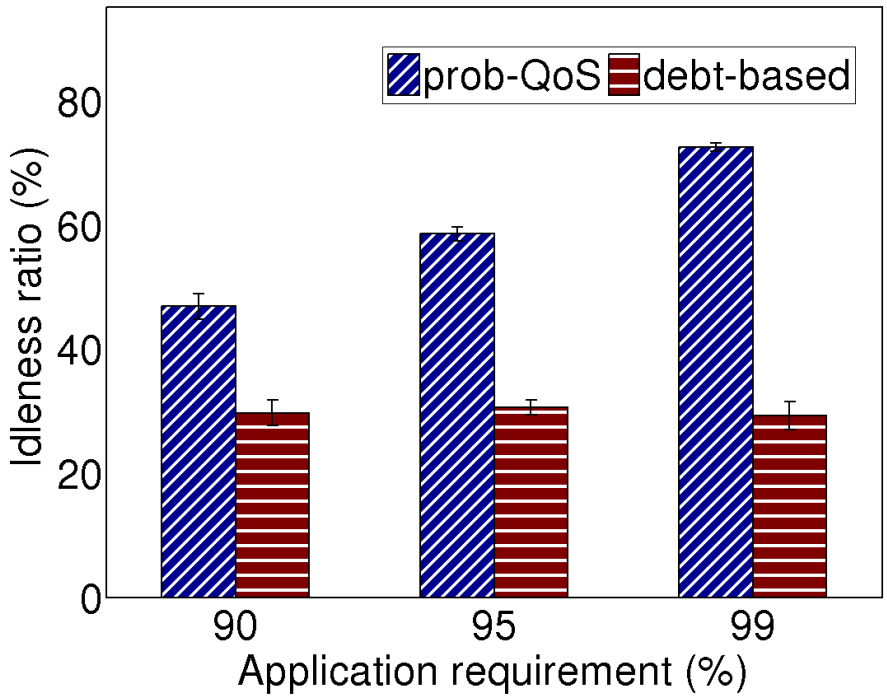


Figure 4.6: Idleness ratio: prob-QoS vs debt-based algorithm with different application requirements.

4.6 Extensions

The previous sections have focused on centralized schemes to ensure per-packet real-time communication guarantees in single-channel settings. In this section, we discuss the impact of the timescale of real-time communication guarantees on system throughput, how to leverage multiple wireless channels, how to address the impact of control signaling unreliability, and how to implement the proposed algorithms in a distributed manner.

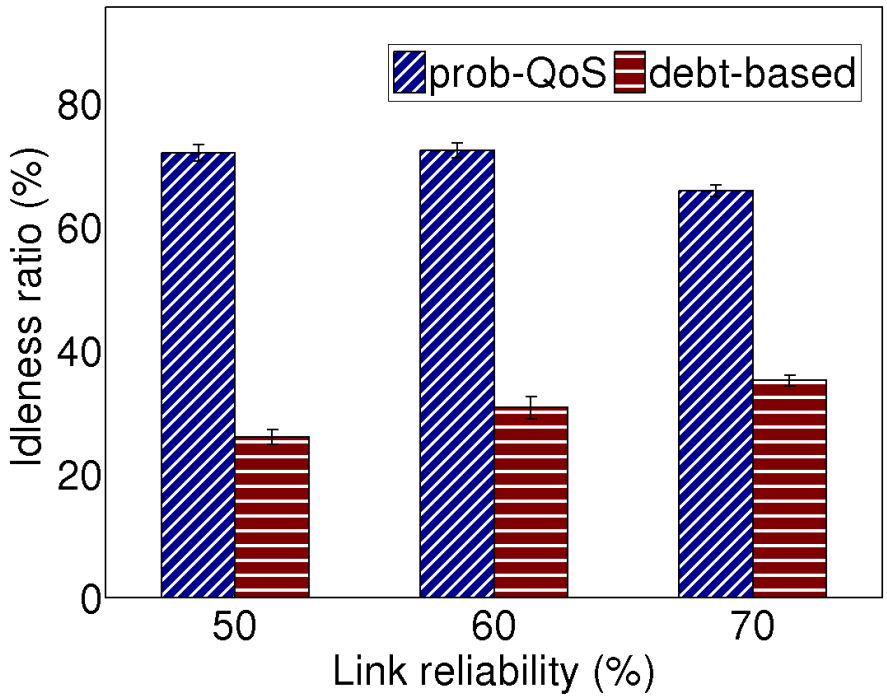


Figure 4.7: Idleness ratio: prob-QoS vs debt-based algorithm with different link reliability.

4.6.1 Impact of timescales of real-time communication guarantee

Targeting applications such as industrial control and augmented reality which are sensitive to the delay and loss of every packet (or every few consecutive packets), the studies in this paper have mainly focused on how to ensure probabilistic short-term, per-packet communication delay bound. Existing studies such as those by Hou et al. [38, 39], on the other hand, have focused on how to ensure a certain long-term, asymptotic average ratio of packets that are delivered before deadlines, instead of ensuring probabilistic real-time delivery of each packet. Therefore, a question worth asking is how the timescales (e.g., short-term vs. long-term) of real-time communication guarantees impact network behavior such as throughput and per-packet delivery reliability.

To answer the above question, here we examine the question of how to schedule the transmissions of packets so that, for each link i, out of every "window" of K_i consecutive packets, the probability $\mathbb{P}_i(m_i, p_i, K_i, r_i)$ for r_i fraction of the K_i packets to be delivered before

deadlines is no less than q_i. In this case, K_i is the "timescale of real-time communication guarantee", and here we intend to understand the impact of K_i on network behavior such as throughput. To facilitate the analysis, we assume that, for each link i, each period has T_i time slots, and, as before, the per-transmission reliability along link i is p_i. Figure 4.8 shows

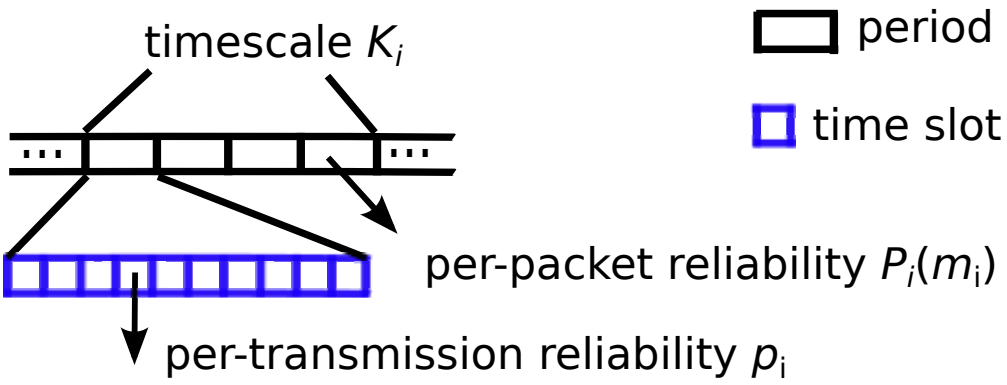


Figure 4.8: Timescale analysis.

the setup for the analysis. (Note that, as special cases of the analytical setup, the problem considered in the previous sections of this paper is such that $K_i = 1$, $r_i = 1$, and $q_i = P_i$, and the problems considered by Hou et al. [38, 39] are such that $K_i = \infty$.)

Given that, with techniques such as power control, the per-transmission reliability can be controlled to be *i.i.d.*, the number of transmission opportunities reserved for each packet is assumed to be the same and denoted by m_i. That is, each packet is transmitted until success or up to m_i times in each period, and $1 \leq m_i \leq T_i$. Let $P_i(m_i) \triangleq Prob\{\text{Per-packet delivery success}\}$, then

$$\begin{aligned} P_i(m_i) &= Prob\{\text{A packet delivered within } m_i \text{ transmissions}\} \\ &= 1 - Prob\{\text{Fail all } m_i \text{ transmissions}\} \\ &= 1 - \underbrace{(1-p_i)(1-p_i)...(1-p_i)}_{m_i} \\ &= 1 - (1-p_i)^{m_i}. \end{aligned} \tag{4.19}$$

Let $\mathbb{P}_i(m_i, p_i, K_i, r_i)$ be the probability that the real-time delivery ratio along link i is at least r_i for every K_i consecutive packets, then

$$\begin{aligned}
&\mathbb{P}_i(m_i, p_i, K_i, r_i) \\
&= Prob\{\text{At least } \lceil K_i r_i \rceil \text{ packets out of every } K_i \text{ packets} \\
&\qquad \text{are delivered before deadlines}\} \\
&= \sum_{K_i'=\lceil K_i r_i \rceil}^{K_i} \text{Prob } \{K_i' \text{ packets out of every } K_i \text{ packets} \\
&\qquad\qquad \text{are delivered before deadlines}\} \\
&= \sum_{K_i'=\lceil K_i r_i \rceil}^{K_i} \binom{K_i}{K_i'} \cdot P_i(m_i)^{K_i'} \cdot (1 - P_i(m_i))^{K_i - K_i'}.
\end{aligned} \tag{4.20}$$

Given that $P_i(m_i)$ and $\mathbb{P}_i(m_i, p_i, K_i, r_i)$ are non-decreasing with m_i, the transmission scheduling scheme should be such that the minimum number of transmission opportunities are reserved for each packet to ensure that $\mathbb{P}_i(m_i, p_i, K_i, r_i) \geq q_i$, if possible. When p_i and T_i are relative small while r_i and q_i are large, it may be infeasible to ensure $\mathbb{P}_i(m_i, p_i, K_i, r_i) \geq q_i$ even with $m_i = T_i$, and the problem is infeasible in this case. For understanding the impact of the timescales of real-time communication guarantee, it is enough for the analysis in this section to focus on cases where the problem is feasible. With this assumption, the transmission scheduling scheme reserves $m_i^*(K_i, p_i, r_i, q_i)$ number of transmission opportunities for each packet, with m_i^* specified as follows:

$$m_i^*(K_i, p_i, r_i, q_i) = \underset{m_i}{\text{argmin}} \ \ \mathbb{P}_i(m_i, p_i, K_i, r_i) \geq q_i. \tag{4.21}$$

It is difficult to derive the close-form, analytical solution for computing $m_i^*(K_i, p_i, r_i, q_i)$, but, given that $\mathbb{P}_i(m_i, p_i, K_i, r_i)$ is non-decreasing with m_i, $m_i^*(K_i, p_i, r_i, q_i)$ can be found by numerically searching from 1, 2, ... until the first m_i such that $\mathbb{P}_i(m_i, p_i, K_i, r_i) \geq q_i$.

To understand the impact of the timescales of real-time communication guarantee (i.e., K_i), Figures 4.9 and 4.10 show the impact of K_i on m_i^* for different parameter config-

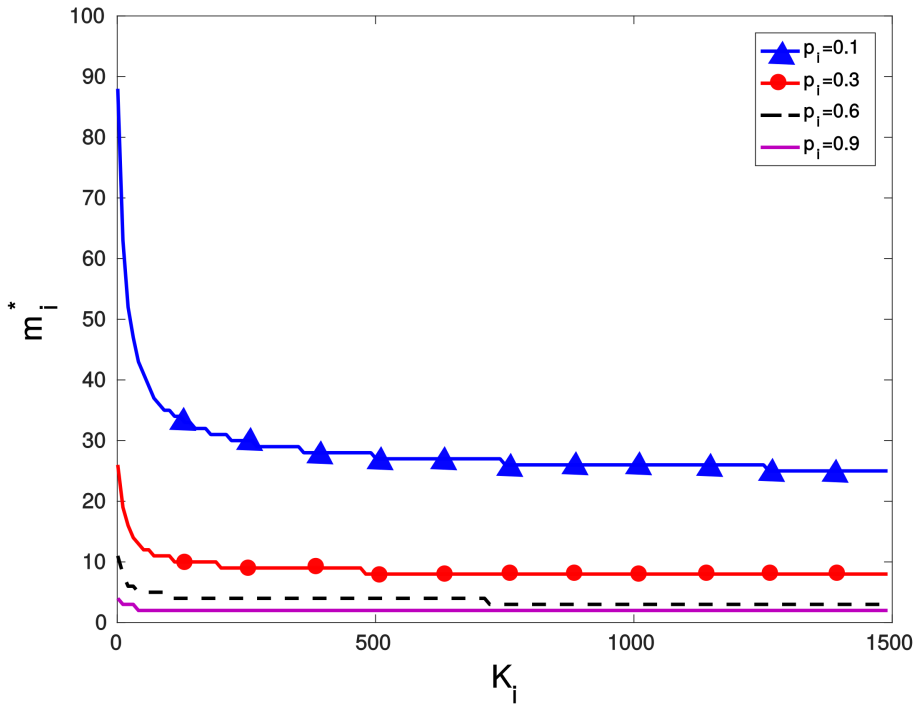


Figure 4.9: An example showing the relation between K_i and m_i^*, where $r_i = 0.9$, $q_i = 0.9999$, and $T_i = 100$.

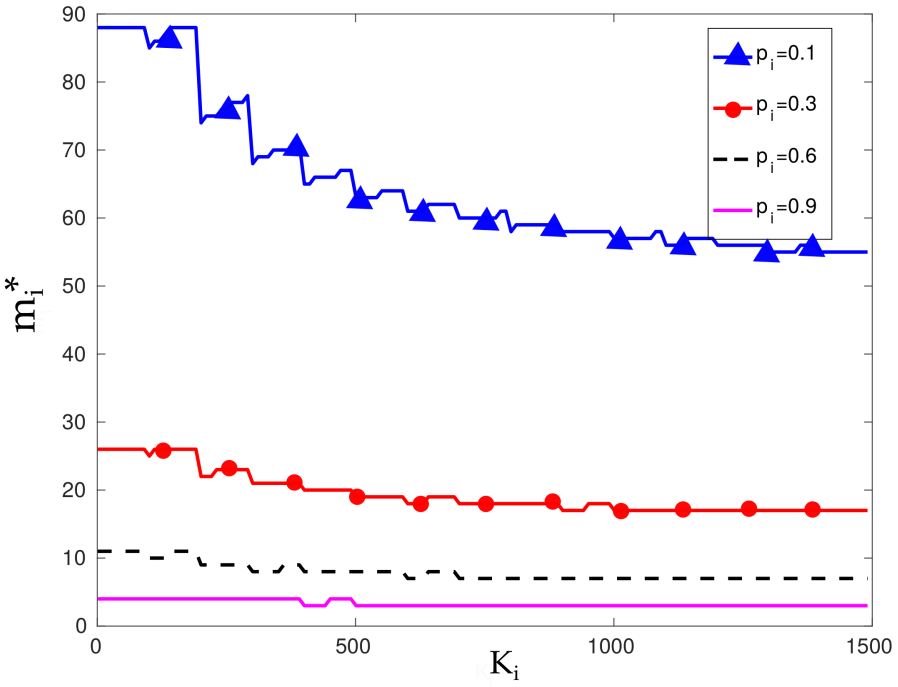


Figure 4.10: An example showing the relation between K_i and m_i^*, where $r_i = 0.99$, $q_i = 0.9999$, and $T_i = 100$.

urations. The figures show that the overall trend is such that m_i^* decreases as K_i increases, even though there exist transient perturbations (i.e., m_i^* increases with K_i) embedded into the overall trend. The overall trend is clearer when the per-transmission reliability p_i is lower, and m_i^* becomes less sensitive to K_i as the per-transmission reliability p_i increases. Given that the supportable network throughput increases with decreasing m_i^*, the observations show that, when the per-transmission reliability is low (e.g., 0.1), the supportable network throughput increases in a non-negligible manner with increasing K_i. On the other hand, as the per-transmission reliability increases, the supportable network throughput becomes less sensitive to K_i. (Similar phenomena have been observed for other parameter configurations, but details are omitted here due to space constraint.)

For a given m_i^* and p_i, there exists a corresponding per-packet success probability P_i^*. Note that, for a given K_i, r_i, and q_i, P_i^* is the same no matter what p_i is; that is, p_i impacts the choice of m_i^* to ensure the required P_i^*. Figures 4.11 and 4.12 show the relation

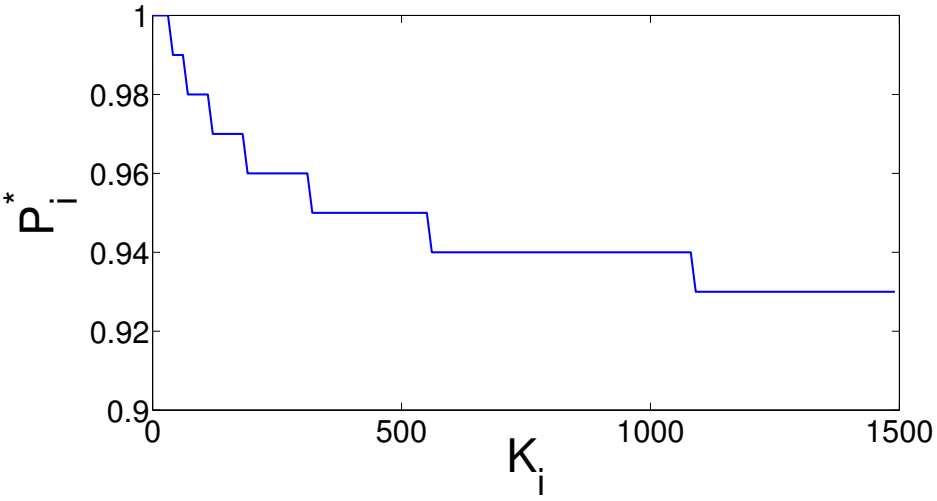


Figure 4.11: An example showing the relation between K_i and P_i^*, where $r_i = 0.9$, $q_i = 0.9999$.

between P_i^* and K_i for the same parameter configurations as those for Figures 4.9 and 4.10 respectively. The figures show that, as K_i increases, P_i^* tends to decrease as an overall trend

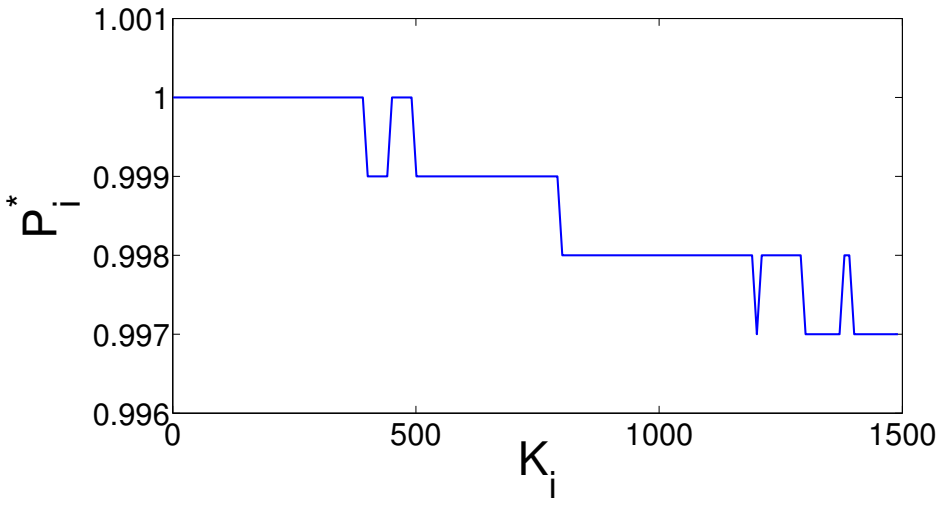


Figure 4.12: An example showing the relation between K_i and P_i, where $r_i = 0.99$, $q_i = 0.9999$.

(despite transient perturbations) and converges to r_i as K_i goes to infinity. When K_i is small, P_i^* tends to be higher than r_i to ensure $\mathbb{P}_i(m_i^*, p_i, K_i, r_i) \geq q_i$.

4.6.2 Extension to multichannel settings

In the previous sections, single channel settings are considered. When multiple channels are available for packet transmissions, the proposed scheme can be extended. The multichannel wireless scheduling problem is similar to the multiprocessor real-time scheduling, with each available channel treated as a processor.

In multiprocessor real-time scheduling, different tasks are assigned to different processors. At any moment in time, there is only one active task on each processor while multiple tasks are running in parallel on different processors. Similarly, in multichannel wireless scheduling, transmission links are assigned to different channels, and within each channel, one link is scheduled to transmit at a time.

Then, by multi-core real-time scheduling theory [58], the P-Fair scheduling algorithm [58] is the optimal, capacity-achieving scheduling algorithm, and the real-time communication requirements of the n mutually-interfering links (see Section 5.3.2) can be satisfied if

the following condition holds:

$$\sum_{i=1}^{n} \rho_i \leq m, \tag{4.22}$$

where ρ_i is the work density of link i as defined by (4.4), and m is the number of available channels. For the usual case when $D_i = T_i$ for every link i (i.e., packet delivery deadline equals packet generation interval), Condition (4.22) is also a necessary condition.

4.6.3 Impact of control signaling unreliability

The presentations in this paper so far have assumed that the delivery of control signaling packets along the downlinks from the AP to sensors is perfectly reliable. When the downlink reliability is less than 1, the loss of a control signal will lead to the situation where no sensor transmits in the corresponding time slot, which can be treated as a packet loss. To deal with this in our framework, we just need to reflect the downlink unreliability in the "adjusted" uplink reliability from each sensor to AP. That is, if p_i^d is the downlink reliability from the AP to sensor i, the "adjusted" uplink reliability becomes $p_i^d * p_i$ for link i, and then the rest of the framework applies.

4.6.4 Distributed solution

Although a centralized network with a base station is considered in this paper, it is possible to extend the solution to a distributed scenario. Assume an *ad hoc* network with multiple sender-receiver pairs. Since the traffic pattern is deterministic for each link (i.e., the traffic period and deadline are known), such information can be shared among links through information exchange before transmissions. As a consequence, each link is aware of the packet arrival and deadline requirement of other interfering links, and then a distributed EDF scheduler can run at each link synchronously [59].

4.7 Concluding remarks

In this paper, a probabilistic framework for per-packet real-time wireless communication guarantees is proposed. The notion of *real-time* in this paper differs fundamentally from

existing work in the sense that it ensures in an *arbitrarily* long execution history, a *randomly* selected packet will meet its deadline in a user-specified probability. By R3 mapping, the upper-layer requirement and the lower-layer link reliability are translated into the number of transmission opportunities needed. By optimal real-time communication scheduling as well as admission test and traffic period optimization, the system utilization is maximized while the schedulability is maintained. The proposed admission test is proved to be both sufficient and necessary. The simulation results show that the proposed scheme can meet the probabilistic real-time communication requirement.

For future work, we plan to integrate the proposed probabilistic real-time framework with PRKS [37], where the link reliability is guaranteed by PRKS and the timeliness is provided by the proposed framework in this paper. Another interesting direction would be to consider aperiodic traffic coexisting with real-time traffic, in which case the channel utilization can potentially be improved by allowing the aperiodic traffic to utilize the time slots that are left idle by the periodic, real-time traffic. We have assumed that the per-packet transmission status (i.e., success or failure) is independent across different links in this work, and a future direction is to consider spatial link correlation and diversity in real-time scheduling which may increase system real-time capacity (i.e., amount of data deliverable within certain deadlines). Detailed study of alternative link models (e.g., when the communication reliability across a link is not i.i.d.) and distributed implementation of the proposed scheduling algorithm is also an interesting future direction. This study has focused on link-level scheduling, and an interesting future extension will be to consider end-to-end real-time guarantees which address the delay introduced by computational tasks and multi-hop communications.

Table 4.1: Comparison of recent studies that consider real-time communication guarantees in wireless networks.

Paper	Model	Algorithm	Applications
[38]	N wireless clients and one access point, homogeneous delay requirement.	Largest Debt First.	VoIP, Video streaming.
[39]	N wireless clients and one access point, heterogeneous delay requirement.	Pseudo-debt-based Scheduling.	VoIP, Video streaming.
[50]	A WirelessHART network consisting of field devices, one gateway, and a network manager.	Branch-and-Bound (Optimal); Conflict-aware Least Laxity First (Heuristic).	WirelessHART.
[51]	A wireless network consisting of a set of field devices, a gateway, and multiple access points.	Earliest Deadline First (EDF).	Wireless Sensor-Actuator Networks (WSANs).
[70]	An IEEE 802.15.4 network operating in beacon-enabled mode and with a star topology.	A distance constrained real-time off-line message scheduling algorithm.	Industrial Wireless Sensor Networks.
[71]	A typical infrastructure WLAN.	QoS-enabled 802.11e scheme.	Industrial networks.
[72]	An IEEE 802.11 network.	A TDMA data link layer protocol based on IEEE 802.11 physical layer to provide deterministic timing on packet delivery.	Wireless Cyber-Physical Control Applications.

Table 4.2: Two cases with the same long-term average timely delivery ratio but different short-term performance.

Case	Period 1	Period 2	Period 3	Period 4
1	90%	90%	90%	90%
2	99%	79%	99%	83%

CHAPTER 5: PROBABILISTIC PER-PACKET REAL-TIME GUARANTEE WITH JITTER CONTROL

5.1 Introduction

Wireless networked sensing and control (WSC) systems typically comprise a controller and multiple spatially distributed sensors and actuators. The sensors sense data periodically from the physical world and send it to the controller. The controller processes the sensing data to generate control actions for the actuators. For WSC, control performance strongly depends on the timely delivery of the sensing information. If not delivered by a certain deadline, it may degrade the performance of the system and even jeopardize its stability. However, the message transmissions in wireless network may experience delay and delay variations (i.e., jitter) due to various factors such as scheduling algorithm, online trafficload changes, channel fading, and environment dynamics.

One interesting problem is how to reduce delay jitter while ensuring that packets are delivered by their deadlines.

5.2 Related Work

The problem of delay jitter control has been studied for many years in real-time systems. In the following, we will briefly review some representative work in this field.

Cervin et al. [81] proposed the concept of *jitter margin* and combined it with real-time theory to guarantee the schedulability and stability of the system. A lower bound under EDF scheduling is found by estimating the worst-case and best-case response time. However, they did not discuss how to achieve the bound.

A indirect way of reducing jitter in real-time systems is to assign tasks tighter deadlines. Based on EDF scheduling, Baruah et al. [76] proposed to assign shorter relative deadlines to a task set such that the jitter is reduced while the schedulability is maintained. The intuition is that, as long as the schedule is feasible, the jitter will be upper bounded by $D_i - C_i$, where D_i and C_i are the relative deadline and worst-case execution time (WCET). A major drawback is that the task set may not be schedulable any more after advancing

deadlines. In practice, this method only works when the system has a low utilization. Working on this line, [93] proposed an adaptive approach to minimize jitter for a set of tasks. The approach formulates the jitter minimization problem as an optimization problem and developed a heuristic to solve the optimization problem. Instead of just reducing deadline, this approach adaptively adjust deadlines to explore the full design space to reduce the overall jitter. However, both methods are offline algorithms based on the WCET. In general, the actual execution time is much less than the WCET. Thus, these algorithms leave the system unnecessarily idle.

Another approach to reduce jitter in real-time systems is to divide each task into three subtasks, i.e., input, processing, and output [83]. The key idea is to always execute the input subtask at the beginning of the period and the output subtask at the end. By applying this method, the input and output subtasks are always separated by exactly one period, leading to fixed input-output delay thus reducing jitter. However, by inserting idleness between the input and the output, not only the average delay is unnecessarily increased, but extra interference is introduced due to more contention at a same time. In addition, such decomposition of a task introduces more complexity and may not be feasible at all in practice.

Another heuristic to reduce scheduling-induced jitter is to disable preemption during tasks execution. The intuition is to minimize the difference between the time a job's start and completion times. One clear drawback of this method is the reduction of schedulability due to non-preemption.

In communication networks, a buffer can be deployed at the receiver side to smooth out delay jitter [90]. Basically, if a packet arrives earlier than its assigned deadline at the receiver, it will be hold for the remaining time and only delivered to the application unitil the deadline is reached. Thus, the delay jitter can be zero. However, such mechanism is fundamentally the same as the task splitting approach, i.e., the delay is always as large as the deadline.

5.3 System Model and Problem Statement

5.3.1 System model

Consider a wireless network with a centralized controller and N sensors. All the sensors are within the communication range of the controller. Time is slotted and synchronized across the controller and sensors, and the wireless transmissions between the sensors and controller are scheduled in a TDMA manner. At the beginning of each time slot, the controller broadcasts a short control signaling message to indicate the sensor that shall transmit in the current time slot. The duration of a time slot is the time required for the controller to send the control packet plus the time for a sensor to transmit a data packet of certain fixed length. All the time slots are of the same length. To avoid transmission collision, there can be at most one active sensor transmitting at each time slot.

The transmission link between the controller and sensor i is characterized by a 4-tuple (T_i, D_i, p_i, P_i) as shown in Figure 5.1.

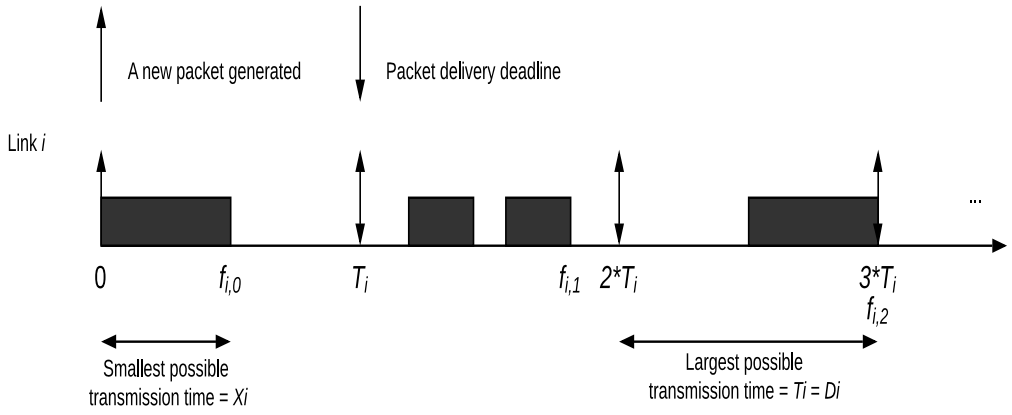


Figure 5.1: Timing for link i.

- T_i: sampling period. Sensor i periodically generates one data packet at time slot $r_{i,k} = kT_i$ $(k = 0, 1, 2, ...)$. In a very general framework, T_i is the design variable to represent the possible choices of the designer at the stage of the system design. Note that in our system model, T_i is a positive integer.

- D_i: relative deadline for sensing data delivery, which is also a positive integer. For sensor i, each packet is associated with a relative deadline D_i in units of time slots. A packet arriving at time slot t should be successfully delivered no later than time slot $t + D_i$; otherwise, the packet is dropped. Since the node always sends new sensing and control signal, $D_i \leq T_i$. In this paper, D_i and T_i are assumed to be equal, and this scenario is usually referred to as *implicit deadline* in classic real-time systems and finds its application in various areas. Note that heterogeneous deadlines across different links are considered since more stringent deadlines are assigned to more critical sensor nodes in practice.
- p_i: link reliability between sensor i and the controller. A packet transmission succeeds in probability p_i. The event of successful transmissions along a link is assumed to be i.i.d. over time, thus whether a packet can be successfully delivered at a time slot is independent of the status of earlier transmissions.
- P_i: application requirement. Due to inherent dynamics and uncertainties in wireless communication, real-time communication guarantees are probabilistic in nature. Thus a probabilistic QoS metric is defined.

  Definition 1 Link i meets probabilistic per-packet real-time guarantee if

$$\forall k, Prob\{f_{i,k} \leq kT_i + D_i\} \geq P_i, \tag{5.1}$$

  where $f_{i,k}$ is the time (measured in time slots) when link i's successfully delivers its k-th packet to the controller.

  Intuitively, the probabilistic per-packet real-time guarantee for link i ensures that in an *arbitrarily* long execution history, a *randomly* selected packet of link i will meet the deadline D_i at least in probability P_i.

5.3.2 Problem statement

Jitter for link i is defined as

$$J_i = \frac{\underset{k}{max}(f_{i,k} - r_{i,k}) - \underset{k}{min}(f_{i,k} - r_{i,k})}{T_i}. \tag{5.2}$$

Intuitively, J_i is the normalized difference between the largest and smallest transmission times among all packets of link i. The overall jitter of all the links is therefore denoted as

$$J = \sum_{i=1}^{N} \frac{J_i}{N}. \tag{5.3}$$

The goal of this paper is to minimize J under the constraint of E.q. (5.1). To answer this question, a probabilistic real-time wireless communication framework is proposed in the next section.

5.4 Offline and Online Scheduling for Probabilistic Per-Packet Real-Time Guarantee with Jitter Control

5.4.1 Overview

An architecture for solving the probabilistic per-packet real-time scheduling problem is shown in Figure 5.2.

First, the “Transmission Opportunities Estimator" uses p_i and P_i of link i to compute X_i, i.e., the number of time slots needed to ensure the successful delivery of a packet along link i in probability P_i. Then the “Deadlines Optimizer" determines whether the links are schedulable or not based on X_i and the input $T_{i,max}$. If schedulable, a set of optimal deadlines $D_{i,OPT}$ are computed based on X_i and T_i. Then the links are scheduled using an optimal real-time scheduling algorithm based on the optimal traffic periods, the transmission opportunities reserved, and delivery deadlines. In the following, we elaborate on each component of the framework.

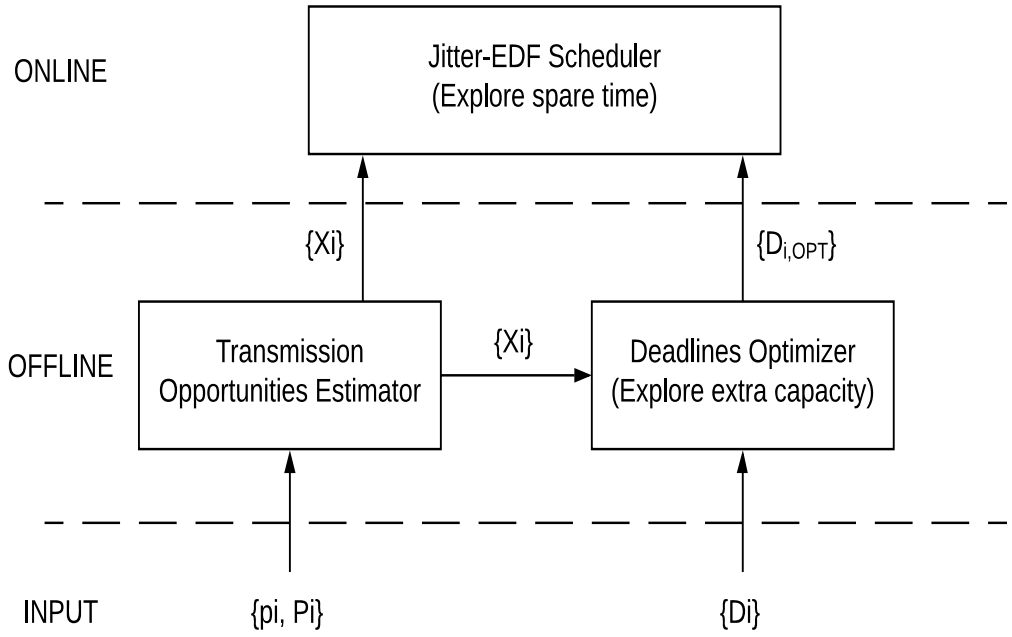


Figure 5.2: Architecture of the proposed framework.

5.4.2 Transmission opportunities estimator

Transmission opportunities estimator is invoked to compute the number of transmission opportunities needed to ensure successful delivery a packet along each link with the required probability guarantee. Given link reliability p_i and application requirement P_i, the minimum number of time slots (or transmission opportunities) needed to ensure successful delivery of a packet for link i in probability P_i, denoted as X_i, can be computed as follows [112]. Let

$$\begin{aligned} Q_i &= Prob\{\text{A packet delivered within } X_i \text{ transmissions}\} \\ &= 1 - Prob\{\text{Fail all } X_i \text{ transmissions}\} \\ &= 1 - \underbrace{(1-p_i)(1-p_i)...(1-p_i)}_{X_i} \\ &= 1 - (1-p_i)^{X_i}. \end{aligned} \tag{5.4}$$

To achieve the application requirement P_i,

$$Q_i = 1 - (1 - p_i)^{X_i} \geq P_i. \tag{5.5}$$

Thus, the minimum number of transmissions needed is

$$X_i = \lceil log_{1-p_i}(1 - P_i) \rceil. \tag{5.6}$$

Note that the X_i reserved time slots do not have to be X_i consecutive time slots, and, for real-time packet delivery, the X_i time slots only have to be before the delivery deadline of the packet.

5.4.3 Deadlines optimizer

Given the better resource utilization and agility that a dynamic-priority algorithm can achieve, in this paper, a dynamic-priority scheduling algorithm based on the Earliest Deadline First (EDF) concept is used. This can be represented by Algorithm 8 [112].

Algorithm 8 Default scheduling algorithm

```
1: for each time slot
2:     target_link_index = 0;
3:     min_deadline = ∞;
4:     for i = 1 to N
5:         if link[i].deadline < min_deadline
6:             min_deadline = link[i].deadline;
7:             target_link_index = i;
8:         end if
9:     end for
10:    Schedule link[target_link_index];
11: end for
```

Theorem 5.4.1. *A set of N links with implicit deadlines is schedulable under Algorithm 8 if and only if*

$$U = \sum_{i=1}^{N} \frac{X_i}{T_i} \leq 1, \tag{5.7}$$

where U is the total utilization of the links.

Definition 2 (Extra capacity) If a set of links has a total utilization $U < 1$, it is said to have an extra capacity of $\Delta U = 1 - U$.

The goal of the deadlines optimizer is to find a set of new deadlines (i.e., periods) for the links when $U < 1$, so as to explore the extra capacity. An optimization problem is formed and solved.

As the first step of optimization, we derive the objective function. Let's revisit E.q. (5.2). Since a packet delivered after its deadline will be dropped, the largest transmission time is its relative deadline D_i. On the other hand, the smallest transmission time is equal to its number of transmission times X_i. Thus,

$$J_i = \frac{D_i - X_i}{T_i}. \tag{5.8}$$

Given the assumption that $T_i = D_i$, the above equation can be rewritten as

$$J_i = 1 - \frac{X_i}{D_i}. \tag{5.9}$$

By inserting E.q. (5.9) into E.q. (5.3), we have

$$J = \sum_{i=1}^{N} \frac{1 - \frac{X_i}{D_i}}{N}. \tag{5.10}$$

Then the problem becomes the following:

$$\underset{d_i, i=1...n}{\text{Minimize}} \quad \sum_{i=1}^{N} \frac{1 - \frac{X_i}{d_i}}{N}, \tag{5.11}$$

subject to:

$$\sum_{i=1}^{N} \frac{X_i}{d_i} \leq 1, \tag{5.12}$$

$$d_i \leq D_i, \quad i = 1, ..., N. \tag{5.13}$$

Basically, the deadlines optimizer finds a set of more stringent deadlines $\{d_i\}$ to minimize the jitter while maintaining the schedulability. This problem has similar structure with [112], thus can be solved accordingly. After the processing of this component, the extra capacity is leveraged. So far, the offline treatment of the links has been finished.

5.4.4 Jitter-EDF scheduler

The transmission opportunities estimator reserves X_i time slots for link i such that the packet is guaranteed to deliver before the deadline at a high probability. However, such reservation scheme may introduce certain pessimism. Due to the probabilistic nature of wireless communication, it may take fewer time slots than X_i for a packet to be delivered.

Definition 3 (Spare time) If the k-th packet along link i tasks less than X_i time slots to be successfully delivered, the remaining time slots reserved in the k-th period for link i is called spare time.

An interesting question is then: when there is spare time, which link will take it? Consider an example with three links. Assume that after the offline processing, the numbers of transmission opportunities and optimal deadlines are obtained as follows: $X_1 = 3$, $D_1 = T_1 = 8$, $X_2 = 6$, $D_2 = T_2 = 16$, $X_3 = 8$, $D_3 = T_3 = 32$. As shown in Figure 5.3 and Figure 5.4, link 1 only takes 2 time slots to successfully deliver its first packet, rather than 3 time slots. Therefore, we say there is one time slot spare time from time 2 to 8. When spare time is available, the scheduler decides which link is going to take it.

Figure 5.3 shows an instance of the default EDF scheduling. At every time slot, the EDF scheduler always schedules the active link who has the earliest absolute deadline. Thus, when there is a spare slot at time 2, link 2 will take it. At time 19, since link 2 and link 3 have a same deadline, i.e., 32, EDF will arbitrarily schedules a link between the two. At time 24, only link 1 and link 2 are active and they have a same deadline. Again, EDF will schedule either of them arbitrarily. As shown in this figure, the total jitter is $J_1 + J_2 + J_3 = 14/16$.

Figure 5.4 shows the results of Jitter-EDF scheduling. The total jitter in this case is 9/16. Therefore, it has a smaller jitter than the default EDF.

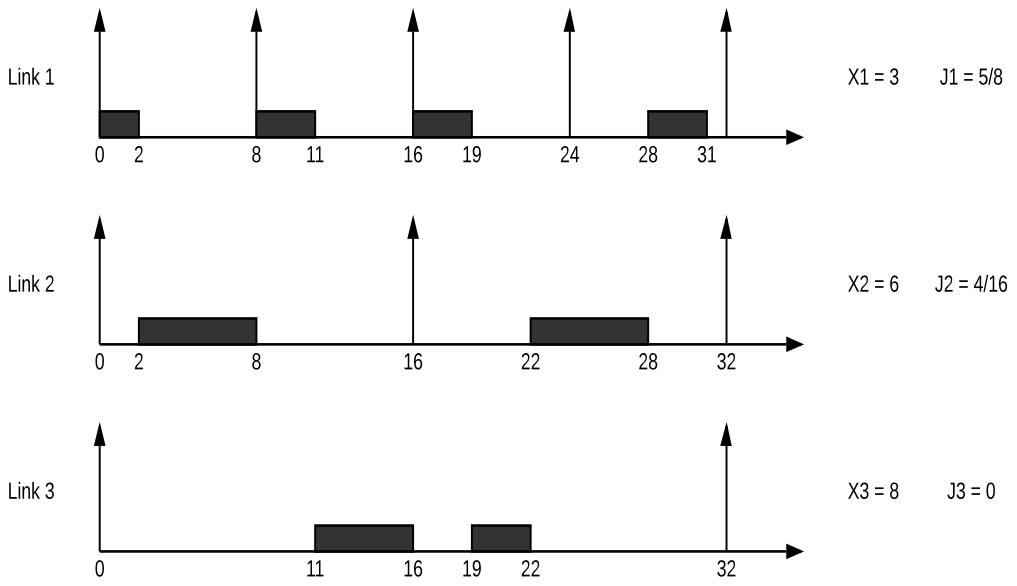


Figure 5.3: Default EDF.

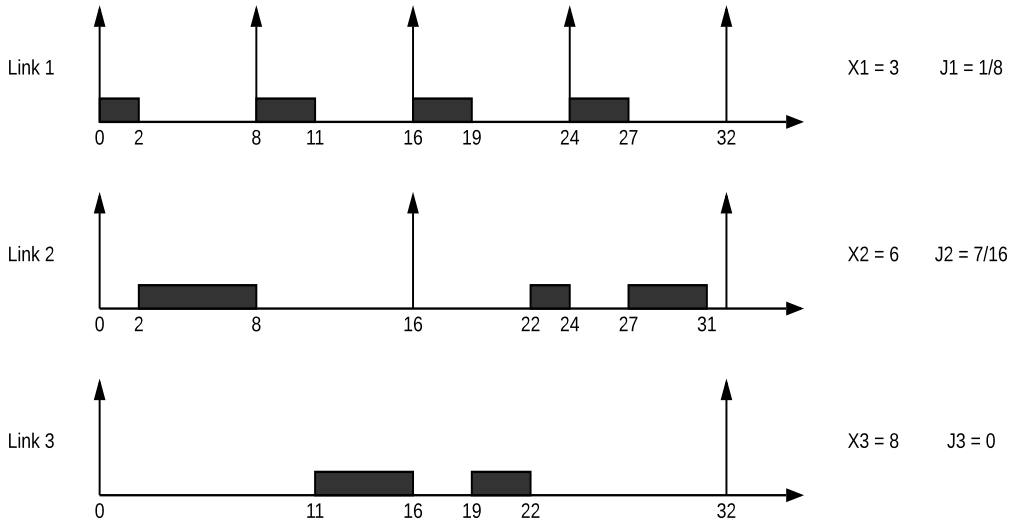


Figure 5.4: Jitter-EDF.

The reason is, though EDF guarantees to meet deadline requirement, it does not consider jitter by design. In the worst scenario, EDF may pick an execution path that leads to the largest jitter (as shown in Figure 5.3) even though 100% schedulability has been achieved. This motivates us to consider the order of transmissions when the time windows till deadlines of multiple links include the spare time slot(s).

Definition 4 (Remaining transmissions) For link i's k-th packet, the remaining transmissions at time slot t is defined as the difference between X_i (i.e., the reserved transmission opportunities for the k-th packet) and the number of transmissions link i has tried from time slot $k * T_i$ to t.

Based on this, we propose a scheduling algorithm called Jitter-EDF as shown in Algorithm 9. The algorithm gives the link with least remaining transmissions the highest priority when multiple links have the same deadline. By doing this, the EDF property is maintained to guarantee schedulability while the jitter is also reduced. The intuition is, by scheduling the link that has least remaining transmissions first, that link can finish its transmissions quickly and quit the contention for wireless channel. Thus, the interference can be reduced for the whole system.

5.5 Simulation

In this section, the performance for the proposed Jitter-EDF is evaluated via Matlab simulation.

5.5.1 Simulation setup

Two scenarios are considered: the first one has a small number of links and the other has more links. Detailed parameters are shown in Table 5.3. All the periods and transmission opportunities are integers. The tilization is guaranteed to be less than or equal to 1. The case that the set of links is not feasible is not within the scope of this paper. The default EDF algorithm is used as a baseline algorithm. In all cases, the simulation is repeated for 100 times. The average jitter is calculated for each case.

Algorithm 9 Jitter-EDF scheduling algorithm

```
1: for each time slot
2:     min_deadline = ∞;
3:     candidates = {};
4:     for i = 1 to N
5:         if link[i].deadline < min_deadline
6:             min_deadline = link[i].deadline;
7:         end if
8:     end for
9:     for i = 1 to N
10:        if link[i].deadline == min_deadline
11:            candidates = candidates ∪ i;
12:        end if
13:    end for
14:    if candidates.size == 1
15:        Schedule the only link in candidates set;
16:    else
17:        Schedule the link in candidates set that has the least remaining transmissions;
18:    end if
19: end for
```

Table 5.3: Simulation setup.

# of links	Periods	Transmission Opportunities
3	{8, 16, 32}	{3, 6, 8}
10	{16, 17, 18, 20, 27, 34, 40, 50, 100, 150}	{2, 3, 2, 2, 2, 3, 3, 2, 4, 4}

5.5.2 Simulation results

Figure 5.5 compares the overall jitter of our algorithm with EDF when there are 3 links. Bar graph with 95% confidence interval is drawn. The result shows our algorithm can significantly reduce the jitter compared to the default EDF. In addition, EDF has a larger variation in jitter compared to our algorithm. The reason is, EDF does not consider the order of concurrent links who have the same deadline. On the contrary, our algorithm always pick the one that has the least remaining transmission opportunities first, thus is more deterministic.

Figure 5.6 shows the result when the number of links increases to 10. As shown in the figure, our algorithm has better performance than default EDF in terms of average jitter.

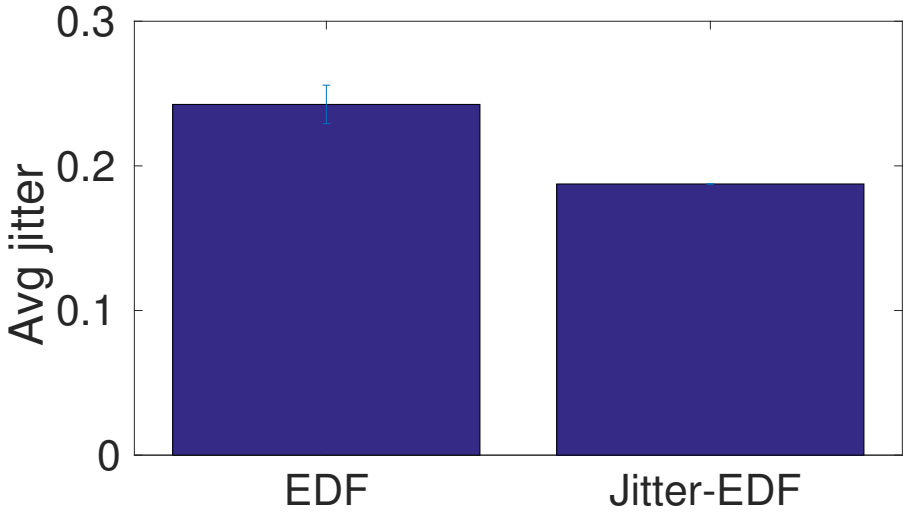


Figure 5.5: Jitter-EDF vs EDF for the case of 3 links.

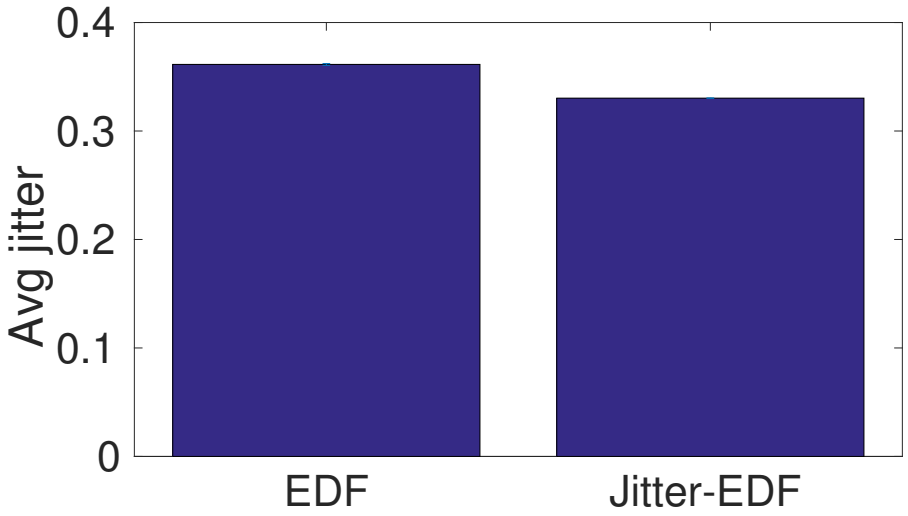


Figure 5.6: Jitter-EDF vs EDF for the case of 10 links.

5.6 Concluding Remarks

In this paper, we have aimed to minimize jitter while guaranteeing deadlines of packets delivery. Our proposed solution consists of several building blocks: a transmission opportunities estimator to reserve enough time slots for each link, a deadline optimizer to further reduce deadlines so as to reduce the jitter whenever possible, and a jitter-aware EDF that considers the transmission order when multiple links have a same deadline. The simulation results show that our method performs better than the default EDF in terms of average jitter.

In this study, we assume that each packet is generated at the beginning of each period. For future work, we plan to investigate the case when packets are released in an asynchronous way. In this case, the jitter could potentially be reduced by carefully decide the release times of each packet.

This study has focused on single-hop transmission. An interesting future extension will be to consider end-to-end jitter reduction where each hop may introduce different degrees of delay variation.

CHAPTER 6: CONCLUSION

In this Ph.D. dissertation, we presented our research accomplishments to achieve real-time guarantees for wireless networked sensing and control.

6.1 Summary of Contributions

In Chapter 2, we present a maximal concurrency and low latency distributed scheduling protocol called ONAMA (Optimal Node Activation Multiple Access). It activates as many nodes as possible while ensuring transmission reliability (in terms of packets delivery ratio). We implemented and tested ONAMA on our NetEye testbed at Wayne State University as well as another testbed at NUS in Singapore. The results of both testbeds shows that ONAMA can maximize throughput while maintaining reliability.

In Chapter 3, we propose algorithms to address the problem of clustering heterogeneous reliability requirements into a limit set of service levels. We formally prove that our solutions are optimal, and they also provide guaranteed reliability.

In Chapter 4, a probabilistic framework for per-packet real-time wireless communication guarantees is proposed. The per-packet probabilistic real-time QoS was formally modeled. By R3 mapping, the upper-layer requirement and the lower-layer link reliability are translated into the number of transmission opportunities needed. By optimal real-time communication scheduling as well as admission test and traffic period optimization, the system utilization is maximized while the schedulability is maintained.

In Chapter 5, we study the problem of jitter control for wireless real-time packets delivery. Our solution is a two-level algorithm. The offline component optimizes the deadlines to explore the extra capacity while the online component further reduces the jitter by carefully considering the transmission orders of different links.

6.2 Future Research Directions

We plan to explore the following directions:

First, our study in this dissertation has focused on single-hop scenario. An interesting future extension will be to consider end-to-end real-time guarantee where each hop may introduce different delay and jitter.

Second, we plan to integrate our proposed framework and algorithms with PRKS, where the link reliability is guaranteed by PRKS and the timeliness is provided by the components proposed in this paper.

Third, we would like to consider aperiodic traffic coexisting with real-time traffic, in which case the channel utilization can potentially be improved by allowing the aperiodic traffic to utilize the time slots that are left idle by the periodic, real-time traffic.

We hope our work will shed lights for future research in mission-critical wireless networked sensing and control system.

APPENDIX

Journal Publications

J1. Y. Chen, H. Zhang, N. Fisher, L. Y. Wang, G. Yin, "Probabilistic Per-Packet Real-Time Guarantees for Wireless Networked Sensing and Control." *IEEE Transactions on Industrial Informatics*, accepted for publication.

J2. H. Zhang, X. Liu, C. Li, Y. Chen, X. Che, F. Lin, L. Y. Wang, G. Yin, "Scheduling with Predictable Link Reliability for Wireless Networked Control." *IEEE Transactions on Wireless Communications (TWC)*, accepted for publication.

J3. Y. Wang, Y. Chen, C. Li, H. Zhang, J. Rao, P. Gossman, J. Zhu, "VInsight: Enabling Open Innovation in Networked Vehicle Sensing and Control." *IEEE Network*, 30 (4), 34-44, 2016.

J4. X. Liu, Y. Chen, H. Zhang, "A Maximal Concurrency and Low Latency Distributed Scheduling Protocol for Wireless Sensor Networks." *International Journal of Distributed Sensor Networks (Hindawi)*, 2015 (invited paper).

Conference Publications

C1. Y. Chen, H. Zhang, "Optimal Request Clustering for Link Reliability Guarantee in Wireless Networked Control." *IEEE Wireless Communications and Networking Conference (WCNC)*, 2017.

C2. H. Zhang, X. Liu, C. Li, Y. Chen, X. Che, F. Lin, L. Y. Wang, G. Yin, "Scheduling with Predictable Link Reliability for Wireless Networked Control." *IEEE/ACMInternational Symposium on Quality of Service (IWQoS)*, 2015.

Posters

P1. H. Zhang, C. Li, Y. Chen, P. Ren, L. Wang, "Predictable Wireless Networking for Real-Time Cyber-Physical-Human Systems." *ACM/IEEE International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2017.

P2. H. Zhang, J. Hua, S. Shu, H. Jin, C. Li, Y. Chen, "Predictable Wireless Networking and Collaborative 3D Vision for Real-Time Cyber-Physical-Human (CPH) Systems." *US-Ignite Conference*, 2017.

P3. H. Zhang, X. Liu, C. Li, Y. Chen, X. Che, F. Lin, L. Y. Wang, G. Yin, "PRK-Based Scheduling for Predictable Link Reliability in Wireless Networked Sensing and Control." *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2013.

REFERENCES

[1] M. Doddavenkatappa, M. C. Chan, and A. L. Ananda. Indriya: A low-cost, 3D wireless sensor network testbed. *Testbeds and Research Infrastructure. Development of Networks and Communities*, 302-316, 2012.

[2] TelosB sensor node. `http://www.memsic.com`.

[3] TinyOS. `http://www.tinyos.net/`.

[4] X. Ju, H. Zhang, and D. Sakamuri. NetEye: a user-centered wireless sensor network testbed for high-fidelity, robust experimentation. *International Journal of Communication Systems*, 25(9):1213-1229, 2012.

[5] Algorithm "Fast MIS V2". `http://dcg.ethz.ch/lectures/podc_allstars/lecture/podc.pdf`, 2013.

[6] L. Bao and J. J. Garcia-Luna-Aceves. A new approach to channel access scheduling for Ad Hoc networks. In *ACM MobiCom*, 2001.

[7] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu. Impact of interference on multi-hop wireless network performance. *Wireless networks*, 11(4):471-487, 2005.

[8] C. Zhu and M. S. Corson. A five-phase reservation protocol (FPRP) for mobile ad hoc networks. *Wireless Networks*, 7(4):371-384, 2001.

[9] I. Rhee, A. Warrier, J. Min, and L. Xu. DRAND: Distributed Randomized TDMA Scheduling for Wireless Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, 8(10):1384-1396, 2009.

[10] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *ACM SenSys*, 2004.

[11] W. Ye and W. Wang. An energy-efficient MAC protocol for wireless sensor networks. In *IEEE INFOCOM*, 2002.

[12] M. Sha, G. Xing, G. Zhou, S. Liu, and X. Wang. C-mac: model-driven concurrent medium access control for wireless sensor networks. In *IEEE INFOCOM*, 2009.

[13] G. Zhou, T. He, J. A. Stankovic, and T. Abdelzaher. RID: radio interference detection in wireless sensor networks. In *IEEE INFOCOM*, 2005.

[14] J. Ni, B. Tan, and R. Srikant. Q-CSMA: Queue-length-based CSMA/CA algorithms for achieving maximum throughput and low delay in wireless networks. *IEEE/ACM Transactions on Networking*, 20(3):825-836, 2012.

[15] M. Yves, J. M. Robson, S. D. Nasser, and A. Zemmari. An optimal bit complexity randomized distributed MIS algorithm. *Structural Information and Communication Complexity*, 323-337, 2010.

[16] L. B. L. Bao and J. Garcia-Luna-Aceves. Hybrid channel access scheduling in ad hoc networks. In *IEEE ICNP*, 2002.

[17] X. Che, H. Zhang, and X. Ju. The case for addressing the ordering effect in interference-limited wireless scheduling. *IEEE Transactions on Wireless Communications*, 13(9):5028-5042, 2014.

[18] ISA SP100.11a. `http://www.isa.org//MSTemplate.cfm?MicrositeID=1134&CommitteeID=6891`.

[19] WirelessHART. `http://www.hartcomm.org/protocol/wihart/wireless_technology.html`.

[20] J. Baillieul and P. J. Antsaklis. Control and communication challenges in networked real-time systems. *Proceedings of the IEEE*, 95(1):9-28, 2007.

[21] M. Pajic, S. Sundaram, G. J. Pappas, and R. Mangharam. The wireless control network: a new approach for control over networks. *IEEE Transactions on Automatic Control*, 56(10):2305-2318, 2011.

[22] S. Han, X. Zhu, A. K. Mok, D. Chen, M. Lucas, and M. Nixon. Reliable and real-time communication in industrial wireless mesh networks. in *IEEE RTAS*, 2011.

[23] A. Willig. Recent and emerging topics in wireless industrial communications: A selection. *IEEE Transactions on Industrial Informatics*, 4(2):102-124, 2008.

[24] H. Zhang, X. Liu, C. Li, Y. Chen, X. Che, F. Lin, L. Y. Wang, and G. Yin. Scheduling with predictable link reliability for wireless networked control. In *IEEE IWQoS*, 2015.

[25] H. Zhang, X. Che, X. Liu, and X. Ju. Adaptive instantiation of the protocol interference model in wireless networked sensing and control. *ACM Transactions on Sensor Networks*, 10(2), 2014.

[26] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams: theory and practice. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):515-528, 2003.

[27] S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129-137, 1982.

[28] D. Arthur and S. Vassilvitskii. k-means++: the advantages of careful seeding. In *ACM SODA*, 2007.

[29] B. Jackson, J. D. Scargle, D. Barnes, S. Arabhi, A. Alt, P. Gioumousis, E. Gwin, P. Sangtrakulcharoen, L. Tan, and T. T. Tsai. An algorithm for optimal partitioning of data on an interval. *IEEE Signal Processing Letters*, 12(2):105-8, 2005.

[30] J. D. Scargle and M. K. Quweider. Edge detection using dynamic optimal partitioning. In *IEEE ICASSP*, 2006.

[31] R. Zurawski. Industrial communication technology handbook. *CRC Press*, 2014.

[32] I. Hou and P. R. Kumar. Scheduling periodic real-time tasks with heterogeneous reward requirements. In *IEEE RTSS*, 2011.

[33] T. Carley, M. A. Ba, R. Barua, and D. B. Stewart. Contention-free periodic message scheduler medium access control in wireless sensor/actuator networks. In *IEEE RTSS*, 2003.

[34] G. Alderisi, S. Girs, L. Lo Bello, E. Uhlemann, and M. BjÃűrkman. Probabilistic scheduling and Adaptive Relaying for WirelessHART networks. *IEEE ETFA*, 2015.

[35] M. Short and J. Proenza. Towards efficient probabilistic scheduling guarantees for real-time systems subject to random errors and random bursts of errors. *IEEE ECRTS*, 2013.

[36] H. Zhang, X. Che, X. Liu, and X. Ju. Adaptive instantiation of the protocol interference model in wireless networked sensing and control. *ACM Transactions on Sensor Networks*, 10(2), 2014.

[37] H. Zhang, X. Liu, C. Li, Y. Chen, X. Che, F. Lin, L. Y. Wang, and G. Yin. Scheduling with predictable link reliability for wireless networked control. In *IEEE IWQoS*, 2015.

[38] I. Hou, V. Borkar, and P. R. Kumar. A theory of QoS for Wireless. In *INFOCOM*, 2009.

[39] I. Hou, and P. R. Kumar. Scheduling heterogeneous real-time traffic over fading wireless channels. In *INFOCOM*, 2010.

[40] A. Gosain, M. Berman, M. Brinn, T. Mitchell, C. Li, Y. Wang, H. Jin, J. Hua, and H. Zhang. Enabling campus edge computing using GENI racks and mobile resources. In *ACM/IEEE SEC*, 2016.

[41] R. Zurawski (Editor). Industrial Communication Technology Handbook, CRC Press, 2015.

[42] S. Boyd, and L. Vandenberghe. Convex Optimization, Cambridge University Press New York, NY, USA, 2004.

[43] Y. Wu, G. Buttazzo, E. Bini, and A. Cervin. Parameter selection for real-time controllers in resource-constrained systems. *IEEE Transactions on Industrial Informatics*, 6(4), pp. 610-620, 2010.

[44] C. L. Liu, and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1), 1973.

[45] J. Leung, and J. W. Whitehead. On the complexity of fixed priority scheduling of periodic real-time tasks. *Performance Evaluation*, 2(4), 1982.

[46] X. Zhang, and M. Haenggi. Delay-optimal power control policies. *IEEE Transactions on Wireless Communications*, 11(10), 2012.

[47] H. Shariatmadari et al. Machine-type communications: current status and future perspectives toward 5G systems. *IEEE Communications Magazine*, 53(9), 2015.

[48] J. Liu. Real-time systems, Prentice Hall PTR, 2000.

[49] H. Aydin, R. Melhem, and D. Mosse. Optimal reward-based scheduling of periodic real-time tasks. In *RTSS*, 1999.

[50] A. Saifullah, Y. Xu, C. Lu, and Y. Chen. Real-time scheduling for WirelessHART networks. In *RTSS*, 2010.

[51] A. Saifullah et al. Analysis of EDF scheduling for wireless sensor-actuator networks. In *IWQoS*, 2014.

[52] H. J. Korber, H. Wattar, and G. Scholl. Modular wireless real-time sensor/actuator network for factory automation applications. *IEEE Transactions on Industrial Informatics*, vol. 3, no. 2, pp. 111-119, May 2007.

[53] S. Vitturi, L. Seno, F. Tramarin, and M. Bertocco. On the rate adaptation techniques of IEEE 802.11 networks for industrial applications. *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 198-208, Feb. 2013.

[54] J. Heo, J. Hong, and Y. Cho. EARQ: energy aware routing for real time and reliable communication in wireless industrial sensor networks. *IEEE Transactions on Industrial Informatics*, vol. 5, no. 1, pp. 3-11, Feb. 2009.

[55] S. K. Ong, M. L. Yuan, and A. Y. C. NEE. Augmented reality applications in manufacturing: a survey. *International journal of production research*, vol. 46, no. 10, pp. 2707-2742, 2008.

[56] S. Henderson and S. Feiner. Exploring the benefits of augmented reality documentation for maintenance and repair. *IEEE transactions on visualization and computer graphics*, vol. 17, no. 10, pp. 1355-1368, 2011.

[57] F. Lamberti et al. Challenges, opportunities, and future trends of emerging techniques for augmented reality-based maintenance. *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 4, pp. 411-421, 2014.

[58] S. Baruah, and M. Bertogna, and G. Buttazzo. Multiprocessor Scheduling for Real-Time Systems. *Springer*, 2015.

[59] T. L. Crenshaw, A. Tirumala, and S. Hoke. A robust implicit access protocol for real-time wireless collaboration. In *ECRTS*, 2005.

[60] G. Yin and C. Tan and L.Y. Wang and C.Z. Xu. Recursive estimation algorithms for power controls of wireless communication networks. *Journal of Control Theory and Applications*, vol. 6, pp. 225-232, 2008.

[61] L. Xu and L.Y. Wang and G. Yin and H.W. Zhang. Communication information structures and contents for enhanced safety of highway vehicle platoons. *IEEE Transactions on Vehicular Technology*, vol. 63, pp. 4206-4220, 2015.

[62] L.Y. Wang and A. Syed and G. Yin and A. Pandya and H. Zhang. Coordinated Vehicle Platoon Control: Weighted and Constrained Consensus and Communication Network Topologies. In *Proc. 51st Conference on Decision and Control*, 2012.

[63] L.Y. Wang and W. Feng and G. Yin. Joint state and event observers for linear switching systems under irregular sampling. *Automatica*, vol. 49, pp. 894-905, 2013.

[64] F. Wu and G. Yin and L.Y. Wang. Stability of a pure random delay system with two-time-scale Markovian switching. *J. Different. Eqs.*, vol. 253, pp. 878-905, 2012.

[65] G. Yin and Y. Sun and L.Y. Wang. Asymptotic properties of consensus-type algorithms for networked systems with regime-switching topologies. *Automatica*, vol. 47, pp. 1366-1378, 2011.

[66] W. Jiao, M. Sheng, K. S. Lui, and Y. Shi. End-to-end delay distribution analysis for stochastic admission control in multi-hop wireless networks. *IEEE Transactions on Wireless Communications*, 13(3), 2014.

[67] Q. Liu, S. Zhou, and G. B. Giannakis. Cross-layer combing of adaptive modulation and coding with truncated ARQ over wireless links. *IEEE Transactions on Wireless Communications*, 3(5), 2004.

[68] D. Seto, J. P. Lehoczky, and L. Sha. Task period selection and schedulability in real-time systems. In *RTSS*, 1998.

[69] E. Bini and M. D. Natale. Optimal task rate selection in fixed priority system. In *RTSS*, 2005.

[70] S. Yoo et al. Guaranteeing Real-Time Services for Industrial Wireless Sensor Networks With IEEE 802.15.4 Star Topolgy. *IEEE Transactions on Industrial Electronics*, 57(11), 2010.

[71] G. Cena, L. Seno, A. Valenzano and C. Zunino. On the Performance of IEEE 802.11e Wireless Infrastructures for Soft-Real-Time Industrial Applications. *IEEE Transactions on Industrial Informatics*, 6(3), 2010.

[72] Y. H. Wei. et al. RT-WiFi: Real-Time High-Speed Communication Protocol for Wireless Cyber-Physical Control Applications. In *RTSS*, 2013.

[73] Tarek F Abdelzaher, Shashi Prabh, and Raghu Kiran. On real-time capacity limits of multihop wireless sensor networks. In *Real-Time Systems Symposium, 2004. Proceedings. 25th IEEE International*, pages 359–370. IEEE, 2004.

[74] Gopal Agrawal, Biao Chen, Wei Zhao, and Sadegh Davari. Guaranteeing synchronous message deadlines with the timed token medium access control protocol. *Computers, IEEE Transactions on*, 43(3):327–339, 1994.

[75] Amir Aminifar, Petru Eles, and Zebo Peng. Jfair: a scheduling algorithm to stabilize control applications. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2015 IEEE*, pages 63–72. IEEE, 2015.

[76] Sanjoy Baruah, Giorgio Buttazzo, Sergey Gorinsky, and Giuseppe Lipari. Scheduling periodic task systems to minimize output jitter. In *Real-Time Computing Systems and*

Applications, 1999. RTCSA'99. Sixth International Conference on, pages 62–69. IEEE, 1999.

[77] Sanjoy K Baruah, Deji Chen, and Aloysius K Mok. Jitter concerns in periodic task systems. In *Real-Time Systems Symposium, 1997. Proceedings., The 18th IEEE*, pages 68–77. IEEE, 1997.

[78] Enrico Bini and Giorgio Buttazzo. The space of edf deadlines: the exact region and a convex approximation. *Real-Time Systems*, 41(1):27–51, 2009.

[79] Giorgio Buttazzo and Anton Cervin. Comparative assessment and evaluation of jitter control methods. *RTNSâĂŹ07*, page 163, 2007.

[80] Thomas W Carley, Moussa A Ba, Rajeev Barua, and David B Stewart. Contention-free periodic message scheduler medium access control in wireless sensor/actuator networks. In *RTSS*, pages 298–307, 2003.

[81] Anton Cervin, Bo Lincoln, Johan Eker, and Giorgio Buttazzo. The jitter margin and its application in the design of real-time control systems. *RTCSA*, 2004.

[82] Fernanda Coutinho, Jorge Barreiros, Jose A Fonseca, and Ernesto Costa. Jitter minimization with genetic algorithms. In *Factory Communication Systems, 2000. Proceedings. 2000 IEEE International Workshop on*, pages 267–273. IEEE, 2000.

[83] Alfons Crespo, Ismael Ripoll, and Pedro Albertos. Reducing delays in rt control: the control action interval. In *Proceedings of the 14th IFAC World Congress*, pages 257–262, 1999.

[84] Rene L Cruz. A calculus for network delay. i. network elements in isolation. *Information Theory, IEEE Transactions on*, 37(1):114–131, 1991.

[85] Hamza Dahmouni, André Girard, Mohamed Ouzineb, and Brunilde Sanso. The impact of jitter on traffic flow optimization in communication networks. *Network and Service Management, IEEE Transactions on*, 9(3):279–292, 2012.

[86] Robert I Davis and Nicolas Navet. Traffic shaping to reduce jitter in controller area network (can). *ACM SIGBED Review*, 9(4):37–40, 2012.

[87] Marco Di Natale and John A Stankovic. Scheduling distributed real-time tasks with minimum jitter. *Computers, IEEE Transactions on*, 49(4):303–316, 2000.

[88] Libin Dong, Rami Melhem, and Daniel Mossé. Effect of scheduling jitter on end-to-end delay in tdma protocols. In *Real-Time Computing Systems and Applications, 2000. Proceedings. Seventh International Conference on*, pages 223–230. IEEE, 2000.

[89] Emad Felemban, Chang-Gun Lee, Eylem Ekici, Ryan Boder, and Serdar Vural. Probabilistic qos guarantee in reliability and timeliness domains in wireless sensor networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 4, pages 2646–2657. IEEE, 2005.

[90] N Figuera and J Pasquale. Leave-in-time: A new service discipline for real-time communication in a packet-switching data network. In *Proceedings of ACM SIGCOMMâĂŹ95*, pages 207–218, 1995.

[91] Xueying Guo, Rahul Singh, PR Kumar, and Zhisheng Niu. A high reliability asymptotic approach for packet inter-delivery time optimization in cyber-physical systems. In *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 197–206. ACM, 2015.

[92] Ching-Chih Han, Kwei-Jay Lin, and Chao-Ju Hou. Distance-constrained scheduling and its applications to real-time systems. *Computers, IEEE Transactions on*, 45(7):814–826, 1996.

[93] Shengyan Hong, Xiaobo Sharon Hu, and Michael D Lemmon. Reducing delay jitter of real-time control tasks through adaptive deadline adjustments. In *Real-Time Systems (ECRTS), 2010 22nd Euromicro Conference on*, pages 229–238. IEEE, 2010.

[94] I Hou, PR Kumar, et al. Real-time communication over unreliable wireless links: a theory and its applications. *Wireless Communications, IEEE*, 19(1):48–59, 2012.

[95] Praveen Jayachandran and Tarek Abdelzaher. Delay composition algebra: A reduction-based schedulability algebra for distributed real-time systems. In *Real-Time Systems Symposium, 2008*, pages 259–269. IEEE, 2008.

[96] Quan Leng, Yi-Hung Wei, Song Han, Aloysius K Mok, Wenlong Zhang, and Masayoshi Tomizuka. Improving control performance by minimizing jitter in rt-wifi networks. In *Real-Time Systems Symposium (RTSS), 2014 IEEE*, pages 63–73. IEEE, 2014.

[97] Bin Li, Ruogu Li, and Atilla Eryilmaz. Wireless scheduling design for optimizing both service regularity and mean delay in heavy-traffic regimes. 2013.

[98] Bin Li, Ruogu Li, and Atilla Eryilmaz. Throughput-optimal scheduling design with regular service guarantees in wireless networks. *Networking, IEEE/ACM Transactions on*, 23(5):1542–1552, 2015.

[99] Cong Liu and Jian-Jia Chen. Bursty-interference analysis techniques for analyzing complex real-time task models. In *Real-Time Systems Symposium (RTSS), 2014 IEEE*, pages 173–183. IEEE, 2014.

[100] Manuel Lluesma, Anton Cervin, Patricia Balbastre, Ismael Ripoll, and Alfons Crespo. Jitter evaluation of real-time control systems. In *Embedded and Real-Time Computing Systems and Applications, 2006. Proceedings. 12th IEEE International Conference on*, pages 257–260. IEEE, 2006.

[101] Sorin Manolache, Petru Eles, and Zebo Peng. Schedulability analysis of applications with stochastic task execution times. *ACM Transactions on Embedded Computing Systems (TECS)*, 3(4):706–735, 2004.

[102] Yishay Mansour and Boaz Patt-Shamir. Jitter control in qos networks. *IEEE/ACM Transactions on Networking (TON)*, 9(4):492–502, 2001.

[103] Pau Marti, Josep M Fuertes, Gerhard Fohler, and Krithi Ramamritham. Jitter compensation for real-time control systems. In *Real-Time Systems Symposium, 2001.(RTSS 2001). Proceedings. 22nd IEEE*, pages 39–48. IEEE, 2001.

[104] Jerzy Martyna. Scheduling algorithm for delay and jitter reduction of periodic tasks in real-time systems. *Przeglad Elektrotechniczny*, 87(1):236–239, 2011.

[105] Thomas Nolte, Hans Hansson, and Christer Norström. Minimizing can response-time jitter by message manipulation. In *Real-Time and Embedded Technology and Applications Symposium, 2002. Proceedings. Eighth IEEE*, pages 197–206. IEEE, 2002.

[106] Luca Santinelli and Liliana Cucu-Grosjean. Toward probabilistic real-time calculus. *ACM SIGBED Review*, 8(1):54–61, 2011.

[107] Lui Sha, Shirish S Sathaye, and Jay K Strosnide. Analysis of dual-link networks for real-time applications. *Computers, IEEE Transactions on*, 46(1):1–13, 1997.

[108] Rahul Singh, Xueying Guo, and Panganamala Ramana Kumar. Index policies for optimal mean-variance trade-off of inter-delivery times in real-time sensor networks. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 505–512. IEEE, 2015.

[109] Rajdeep Singh, I-Hong Hou, and P Roshan Kumar. Fluctuation analysis of debt based policies for wireless networks with hard delay constraints. In *INFOCOM, 2014 Proceedings IEEE*, pages 2400–2408. IEEE, 2014.

[110] Ken Tindell. *Adding time-offsets to schedulability analysis*. Citeseer, 1994.

[111] Dinesh C Verma, Hui Zhang, and Domenico Ferrari. Delay jitter control for real-time communication in a packet switching network. In *Communications Software, 1991,'Communications for Distributed Applications and Systems', Proceedings of TRICOMM'91., IEEE Conference on*, pages 35–43. IEEE, 1991.

[112] Yu Chen, Hongwei Zhang, Nathan Fisher, Le Yi Wang, George Yin. Probabilistic Per-Packet Real-Time Guarantees for Wireless Networked Sensing and Control. *IEEE Transactions on Industrial Informatics*, accepted for publication.

ABSTRACT

REAL-TIME GUARANTEES FOR WIRELESS NETWORKED SENSING AND CONTROL

by

YU CHEN

May 2018

Advisor: Dr. Hongwei Zhang, Dr. Nathan Fisher

Major: Computer Science

Degree: Doctor of Philosophy

Wireless networks are increasingly being explored for mission-critical sensing and control in emerging domains such as connected and automated vehicles, Industrial 4.0, and smart city. In wireless networked sensing and control (WSC) systems, reliable and real-time delivery of sensed data plays a crucial role for the control decision since out-of-date information will often be irrelevant and even leads to negative effects to the system. Since WSC differs dramatically from the traditional real-time (RT) systems due to its wireless nature, new design objective and perspective are necessary to achieve real-time guarantees.

First, we proposed Optimal Node Activation Multiple Access (ONAMA) scheduling protocol that activates as many nodes as possible while ensuring transmission reliability (in terms of packets delivery ratio). We implemented and tested ONAMA on two testbeds both with 120+ sensor nodes.

Second, we proposed algorithms to address the problem of clustering heterogeneous reliability requirements into a limit set of service levels. Our solutions are optimal, and they also provide guaranteed reliability, which is critical for wireless sensing and control.

Third, we proposed a probabilistic real-time wireless communication framework that effectively integrates real-time scheduling theory with wireless communication. The per-packet probabilistic real-time QoS was formally modeled. By R3 mapping, the upper-layer requirement and the lower-layer link reliability are translated into the number of transmission opportunities needed. By optimal real-time communication scheduling as well as

admission test and traffic period optimization, the system utilization is maximized while the schedulability is maintained.

Finally, we further investigated the problem of how to minimize delay variation (i.e., jitter) while ensuring that packets are delivered by their deadlines.

AUTOBIOGRAPHICAL STATEMENT

Yu Chen received his B.S. and M.S. degrees in wireless communications from the University of Electronic Science and Technology of China in 2007 and 2011, respectively. He is currently a Ph.D. candidate in the Computer Science Department at Wayne State University. His research interests include wireless networked control systems and cyber-physical systems.