IEEE 802.15.4 WIRELESS SENSOR NETWORKS : GTS SCHEDULING AND SERVICE DIFFERENTIATION

Che Woo Na

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Electrical and Comptuer Engineering

Yaling Yang
Dong Ha
Thomas Hou
Jung-Min Park
Liqing Zhang

August 22, 2011
Blacksburg, Virginia

Keywords: Wireless Sensor Networks, Service Differentiation and Real-time Scheduling

IEEE 802.15.4 WIRELESS SENSOR NETWORKS : GTS SCHEDULING AND SERVICE DIFFERENTIATION

Che Woo Na

ABSTRACT

Recently there has been a growing interest in the use of Low Rate Wireless Personal Area Networks (LR-WPAN) [1] driven by the large number of emerging applications such as home automation, health-care monitoring and environmental surveillance. To fulfill the needs for these emerging applications, IEEE has created a new standard called IEEE 802.15.4 for LR-WPAN, which has been widely accepted as the *de facto* standard for wireless sensor networks. Unlike IEEE 802.11 [2], which was designed for Wireless Local Area Networks (WLAN), it focuses on short range wireless communications. The goal of the IEEE 802.15.4 LR-WPAN is to support low data rate connectivity among wireless sensors with low complexity, cost and power consumption [3]. It specifies two types of network topologies, which are the beacon-enabled star network and the nonbeacon-enabled peer-to-peer network. For the beacon-enabled network, it defines the Guaranteed Time Slot (GTS) to provide real-time guaranteed service for delay-sensitive applications.

In the nonbeacon-enabled network the GTS is reserved time slots such that it is requested, allocated and scheduled to wireless sensors that need guaranteed service for delay-sensitive applications. Existing GTS scheduling algorithms include First-Come-First-Served (FCFS) [1], priority-based [4] and Earliest Deadline First (EDF) [5] methods. Such FCFS and priority-based scheduling methods have critical drawbacks in achieving real-time guarantees. Namely, they fail to satisfy the delay constraints of delay-sensitive transactions. Further, they lead to GTS scarcity and GTS underutilization. On the other hand, the EDF-based scheduling method provides delay guarantee while it does not support delay-sensitive applications where arrival of the first packet has a critical impact on the performance.

To solve these problems, we design the optimal work-conserving GTS Allocation and Scheduling (GAS) algorithm that provides guarantee service for delay-sensitive applications in beacon-enabled networks. Not only does the GAS satisfy the delay constraints of transactions, but also it reduces GTS scarcity and GTS underutilization. Further, it supports delay-sensitive applications where arrival of the first packet has a critical impact on the performance. Through the extensive simulation results, we show that the proposed algorithm outperforms the existing scheduling methods. Our algorithm differs from the existing ones in that it is an on-line scheduling and allocation algorithm and allows transmissions of bursty and periodic transactions with delay constraints even when the network is overloaded.

In the nonbeacon-enabled peer-to-peer network some operating scenarios for rate-sensitive applications arise when one considers wireless video surveillance and target detection applications for wireless sensor networks. To support such rate-sensitive applications in wireless sensor networks, we present a Multirate-based Service Differentiation (MSD) operating in the nonbeacon-enabled peer-to-peer network. Unlike existing priority-based service differentiation models, the MSD defines the independent Virtual Medium Access Controls (VMACs), each of which consists of a transmission queue and the Adaptive Backoff Window Control (ABWC). Since the VMACs serve multiple rate-sensitive flows, it is possible that more than one data frame is collided with each other when their backoff times expire simultaneously. To solve such a virtual collision in the virtual collision domain, we design the Virtual Collision Avoidance Control (VCAC). The ABWC component adjusts the backoff window to reflect the local network state in the local collision domain. The VCAC component prevents virtual collisions and preempts packets with the minimal cost in the virtual collision domain. By analyzing these algorithms, we prove that the ABWC component enables the achieved data rate to converge to the rate requirement and the VCAC component produces a virtual-collision-free schedule to avoid degradation of the achieved data rate. Through the simulation, we validate our analysis and show the MSD outperforms existing algorithms.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

During the last decade there has been an explosive growth of research and development in wireless technologies. Such intensive research has had an significant impact on wireless communications among mobile users since there is no need for fixed infrastructure. Including Wi-Fi [6] and HyperLAN [7], these wireless technologies have focused primarily on the capability to support higher data rates with a relatively wide range of deployment while they have not taken account of designs of low power, cost and rate wireless devices required in industrial automation.

Recent advances in low power microprocessors and radio technologies have enabled low cost, low power and multi-functional wireless sensors for home/industrial automation. Such technological advances have created a growing interest in the use of Low Rate Wireless Personal Area Networks (LR-WPAN) [3]. Typical wireless sensors are equipped with a radio transceiver, a data processing unit, a sensor unit and a microprocessor, acquiring information about the surrounding environment. When deployed in a large area, they automatically form a clustered network or an ad hoc multihop network to communicate with each other.

A large number of emerging applications in the LR-WPAN include home automation, healthcare monitoring, environmental monitoring, military application, and so forth. In the healthcare field wireless sensors can be used to remotely monitor physiological data such as blood pressure or heartbeat of patients, and report to the doctor or the hospital when some data

are updated [8]. In the environmental monitoring field they can be deployed to report environmental data such as water pollution, water level and animal species, and collects data concerning their population and habits [9].

To fulfill the needs for these emerging applications in wireless sensors, IEEE has created a new standard called IEEE 802.15.4 [1] for LR-WPAN. This standard has been widely accepted as the *de facto* standard for wireless sensor networks. The goal of the IEEE 802.15.4 LR-WPAN is to provide ultra-low complexity, cost and power for low data rate wireless connectivity for wireless sensors. Different from IEEE 802.11 [2] designed for WLAN, it focuses on short range wireless communications.

Basically, the IEEE 802.15.4 LR-WPAN defines the characteristics of the PHY and Medium Access Control (MAC) for very low power consumption and low duty cycles. While it has features for supporting rate (or bandwidth) and delay-sensitive applications in centralized networks, it does not define any provision for supporting rate-sensitive applications in decentralized networks. In the following sections we address the characteristics of these two network topologies and problems in supporting delay-sensitive and rate-sensitive applications.

1.1 Guaranteed Service for Delay-sensitive Applications in the Centralized Network

Some applications need dedicated bandwidth to achieve low latency and guaranteed service for delay-sensitive data frames (or transactions). To achieve these low latency and guaranteed service, the IEEE 802.15.4 LR-WPAN defines the Guaranteed Time Slots (GTS), which are assigned to wireless sensors needing guaranteed service for delay-sensitive applications. To provide guaranteed service, the GTS is allocated between two contiguous beacon intervals by

the centralized coordinator, called PAN coordinator. To allocate the GTS, the coordinator synchronizes data communication with its neighboring wireless sensors by sending the beacon frame periodically.

There are three design considerations for providing such a guaranteed service for delay-sensitive applications in the beacon-enabled star network. The first is how many GTSs per beacon interval are assigned. The second is how to schedule these GTSs in the beacon interval to satisfy the delay constraints of the transactions. The third is how to make sure that all transactions meet their delay constraints.

With these design considerations, we design a new GTS Allocation and Scheduling (GAS) algorithm that provides guaranteed service for delay-sensitive applications in the beacon-enables star network. Basically, the GAS consists of three components. The first component is the feasibility examination method making sure that all of the transactions are $feasible$, which means that they meet their delay constraints. For this method, we present both how to examine feasibility of these transactions and how to discard transactions to make sure that they remain feasible.

The second component is the GTS allocation and scheduling method, called Earliest Delay Constraint with Minimum Guaranteed Time Slot (EDCF-mGTS), that assigns the minimum number of GTSs to as many transactions as possible in every beacon interval so that more transactions are served in every beacon interval, as compared with EDF. Namely, it is designed to support delay-sensitive applications where arrival of the first packet has a critical impact on the performance. For this, we design the delay model describing the total delay achieved by each transactions. Based on this delay model and feasibility examination method, we present how to reduce GTS scarcity and GTS underutilization.

The third component is the GTS reallocation method that reassigns the GTS unallocated in the beacon interval. This method enables transactions to completed earlier than before GTS reallocation so that it gives longer contention access periods for best-effort traffic. In

Chapter 3, we discuss the details of the GAS design.

1.2 Service Differentiation for Rate-sensitive Applications in the Decentralized Network

While supporting the guaranteed service for delay-sensitive applications in the centralized network, the IEEE 802.15.4 LR-WPAN defines the decentralized network (or nonbeacon-enabled peer-to-peer network) that enables wireless sensors to communicate with one another within their transmission range. For such a peer-to-peer communication among wireless sensors, it defines the unslotted CSMA/CA channel access mechanism by which these wireless sensors compete with one another to hold the shared channel for communication. The purpose of the unslotted CSMA/CA protocol is to allow for wireless sensors to have fair access to the shared wireless channel.

Basically, such a fair channel access mechanism is implemented by the backoff window that determines backoff times right before those wireless sensors attempt to send data frames. Once a wireless sensor holds the shared channel, the others must wait for the sending node to finish its transmission. If a collision occurs in the local collision domain, then the backoff window increases exponentially and backoffs again so that the contention delay increases while the data rate decreases. Due to the shared nature of the wireless channel, this uniform backoff mechanism makes it impossible to design a service differentiation model for rate-sensitive flows with the rate requirement since a larger backoff window leads to a longer latency.

To serve service differentiation for rate-sensitive applications in the IEEE 802.15.4 LR-WPAN, we design the Multirate Service Differentiation (MSD) model. The MSD is implemented by two components. The first component is the Virtual Medium Access Con-

trol (VMAC) which consists of a priority queue and the Adaptive Backoff Window Conrtol (ABWC). One VMAC entity enqueues/dequeues a rate-sensitive flow and adjusts its backoff window depending on the local network state. There are a set of VMACs that serve various flows at the MAC layer.

Since the VMACs serve multiple rate-sensitive flows, it is possible that more than one data frame is collided with each other when their backoff times expire simultaneously. To solve such a virtual collision in the virtual collision domain, the second component, Virtual Collision Avoidance Control (VCAC), is designed. The VCAC component investigates possible collisions that happen in the virtual collision domain, and adjusts the backoff times of data frames that lead to virtual collisions. In Chapter 3, we discuss the details of the MSD design.

1.3 Contributions

The objective of this dissertation is to provide the optimal GTS allocation and scheduling algorithm for delay-sensitive transactions with delay constraints in beacon-enabled star networks, and the service differentiation model for rate-sensitive flows in nonbeacon-enabled peer-to-peer networks. The main contributions of this dissertation are as follows:

Guaranteed Service for Delay-sensitive Applications in the Centralized Network

- We designed an optimal GTS Allocation and Scheduling (GAS) algorithm that allocates and schedules GTSs for delay-sensitive transactions with delay constraints.
- To analyze how long it takes for delay-sensitive transactions to be received at the coordinator, we designed an analytic model, called Cumulative Delay Model (CDM), which describes a total delay of these transactions. Based on this model, we designed the GTS allocation and scheduling method, called Earliest Delay Constraint with Min-

imum Guaranteed Time Slot (EDCF-mGTS), that assigns the minimum number of GTSs to as many transactions as possible in every beacon interval so that more transactions are served in every beacon interval, as compared with EDF.

- To handle overload condition and ensure that a set of transactions are *feasible*, we formulated the feasibility examination method.
- To reduce the number of unused GTSs, a method of maximizing GTSs has been presented.

Service Differentiation for Rate-sensitive Applications in the Decentralized Network

- We designed the multirate service differentiation model for rate-sensitive flows in the nonbeacon-enabled peer-to-peer network.
- To serve individual flows, we designed the independent MAC entity called VMAC, which serves rate-sensitive flow and controls the backoff window size depending on the local network state. To adjust the backoff window every transmission round, the VMAC entity resorts to the ABWC algorithm.
- To handle collisions in the virtual collision domain, we designed the VCAC incurred by simultaneous expiration of these backoff times.

1.4 Organization of the Dissertation

The reminder of this dissertation is organized as follows. (Will be updated)

In Chapter 2, we give an overview of the IEEE 802.15.4 MAC protocol and how to manage the GTS session in the standard. In the chapter, two network topologies for wireless sensor nodes and the superframe structure used by beacons are presented. In particular, GTS session management including GTS allocation and scheduling methods as well as overload condition handling are discussed.

Chapter 3 presents an optimal GTS allocation and scheduling methods that meet the delay constraints of time-sensitive transactions. An analytic delay model is described to analyze relationship between the expected delay and delay constraint in a transaction. In particular, it is proved that the algorithm is optimal like EDF. In addition, a practical algorithm implementation is discussed. For performance evaluation, various types of simulation scenarios such as bursty, periodic and aperiodic traffic are presented to show that the proposed algorithm significantly outperforms existing algorithms in terms of delay guarantee.

Chapter 4 describes the MSD model and its design approach. Two necessary components, AWBW and VCA are inctorucded to support the MSD model, which is followed by how to design and impelement these two components in the IEEE 802.15.4 MAC layer. Algorithm analysis and simulation results will be desribed and compared with existing algrithms.

Chapter 5 concludes our work and discuss future research directions.

CHAPTER 2

BACKGROUND OF THE IEEE 802.15.4 STANDARD AND GTS SESSION MANAGEMENT

The IEEE 802.15.4 LR-WPAN defines the PHY and MAC for very low-power, low-duty cycle network links. It is designed to deploy long-lived systems with low data rate requirements. In this chapter, we give a brief overview of the IEEE 802.15.4 LR-WPAN standard, which includes the network topology, superframe structure and GTS features. Then, we discuss the GTS session management including GTS session establishment, GTS allocation and scheduling methods, and overload condition handling for delay-sensitive transactions with delay constraints. In this section, we give an overview of the IEEE 802.15.4 standard focusing on the network topology, superframe structure and GTS features.

2.1 Protocol Architecture

The IEEE 802.15.4 LR-WPAN defines two layers, i.e., physical and MAC layers (See Figure 2.1). The physical layer defines the data transmission service and the interface to the physical layer management entity. It manages the radio transceiver and supports functionalities for channel selection, link quality estimation, energy detection measurement and clear channel assessment to assist the channel section. It operates on one of three unlicensed frequency bands, 868.0-868.6 MHz, 902-928 MHz and 2400-2483.5 MHz.

The MAC layer allows the transmission of MAC frames through the use of the physical channel. It offers a management interface and manages access to the physical channel. It also controls frame validation, guaranteed services, node associations and security services.

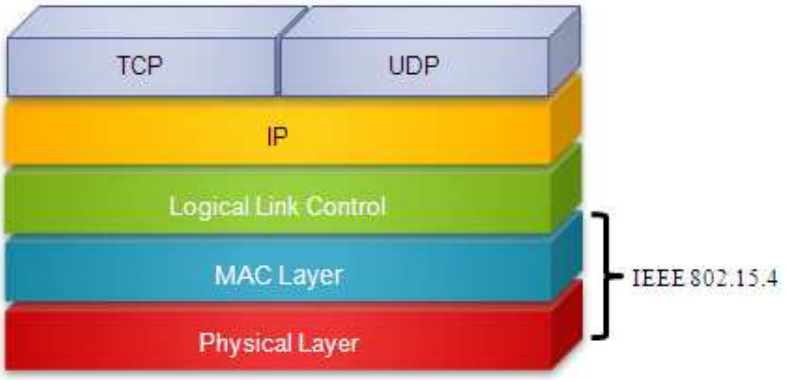


Figure 2.1: IEEE 802.15.4 protocol architecture.

2.2 Network Topology

Depending on the application requirements, the IEEE 802.15.4 defines two topologies: the star topology and the peer-to-peer topology. Figure 2.2 shows the basic structures of these two topologies.

In the star topology, communication links are established between wireless sensors and a single coordinator. The coordinator is used to initiate, terminate or route flows in the network, where a wireless sensor is either the initiation point or the termination point of a flow. Once the coordinator gets started, it allows wireless sensors to join its network and provides two types of medium access to these wireless sensors. One is guaranteed and contention-free access and the other is contention-based access aligned with the boundaries of time slots (slotted CSMA/CA). All star networks operate independently from other star networks.

To communicate with wireless sensors, the coordinator transfers beacon frames to them periodically within the radio coverage so that they are synchronized with these beacon frames.

Once a wireless sensor newly enters a star topology network, it is associated with the coordinator and is able to transmit its data packets.

In the peer-to-peer topology, each wireless sensor is able to communicate with any other wireless sensor within its transmission range. The coordinator is chosen among these wireless sensors and relay data packets to other coordinators. Once it is assigned, it allows only contention-based access (unslotted CSMA/CA) [10] without use of periodic beacons so that all wireless sensors are not synchronized. It is always on and ready to receive data packets from wireless sensors while data transfer in the opposite direction is poll-based. Namely, the wireless sensors periodically wake up and poll the coordinator for the pending data packets. The coordinator then transfer these packets to them.

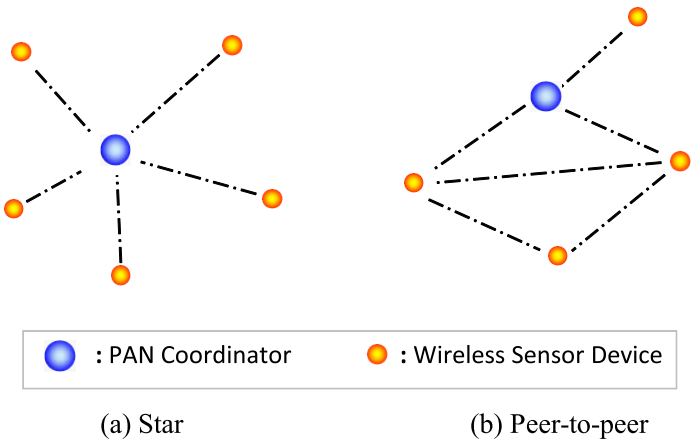


Figure 2.2: IEEE 802.15.4 topology.

2.3 Superframe Structure

In the IEEE 802.15.4 beacon-enabled network, the coordinator maintains a couple of parameters, which are encapsulated into a beacon frame and are transmitted to all wireless sensors within its transmission range. By listening to the beacon frames, the wireless sensors learn their configurations, including $BeaconOrder$ (BO), $Superframe Order$ (SO) and the GTS descriptor. Hereafter, we call these BO and SO beacon parameters.

Figure 2.3 illustrates the superframe structure defined in the IEEE 802.15.4. The superframe is bounded between two consecutive beacons and its duration is described by Superframe Duration (SD) and Beacon Interval (BI)[1], which are given by:

$$\begin{aligned} SD &= \delta \times BaseSuperframeDuration \times 2^{SO}, \\ BI &= \delta \times BaseSuperframeDuration \times 2^{BO}, \end{aligned} \tag{2.1}$$

where $0 \leq SO \leq BO \leq 14$, δ is a symbol period in μs and $BaseSuperframeDuration$ is defined as 960 symbols [1].

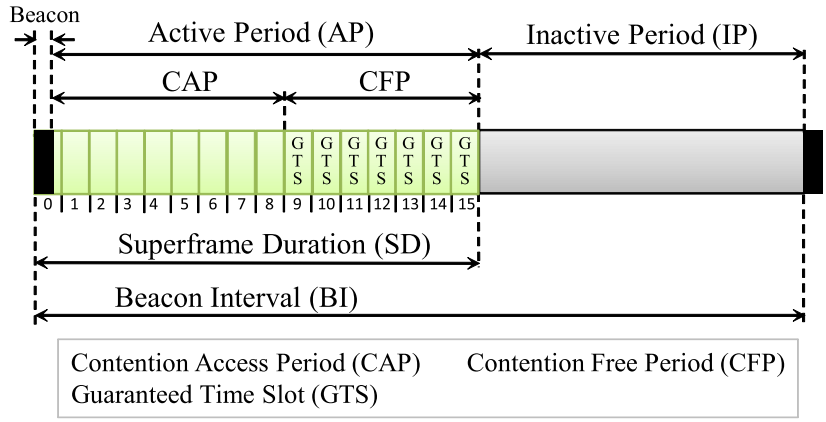


Figure 2.3: The IEEE 802.15.4 superframe structure. A beacon frame has parameters, which describe both the superframe duration and the beacon interval.

The SD includes four parts: a beacon, a Contention Access Period (CAP), a Contention Free Period (CFP) and an Inactive Period (IP). The beacon frame is sent periodically at the starting slot of 0. It contains the addressing fields, superframe specification, GTS descriptors and so forth.

The CAP follows immediately after the beacon and ends at the beginning of the CFP. If the CFP does not exist, the CAP can occupy the entire SD. The minimum length of the CAP is defined as $aMinCAPLength$, which equals 440 symbols. This minimum length ensures that MAC commands are sent when GTSs are used. All transmissions during the CAP are made

[1]In this paper, BI and SD are represented in a time domain, not in the number of symbols.

using slotted CSMA/CA protocol while the acknowledgment of a packet is sent immediately without contention.

The CFP starts right after the CAP and completes before the start of the inactive portion of the superframe. It includes the GTSs allocated by the coordinator and grows or shrinks depending on the total length of all GTSs. Unlike the CAP, the CFP does not use CSMA/CA mechanism to access the channel.

The IP starts right after the SD. During the IP, all wireless sensors enter sleep mode to save their energy. In the IEEE 802.15.4, both the SD and the IP are integral part of a duty cycle. In the following section, we explain how the IEEE 802.15.4 determines a low duty cycle in wireless sensor network.

2.4 CSMA/CA Channel Access Mechanism

The IEEE 802.15.4 LR-WPAN uses two types of channel access mechanism, depending on the network topology. Beacon-enabled PANs use a slotted CSMA/CA channel access mechanism while nonbeacon-enabled PANs use an unslotted CSMA/CA channel access mechanism.

In beacon-enabled PANs, the backoff slots are aligned with the start of the beacon transmission. The backoff slots of all sensor devices within one PAN are aligned to the PAN coordinator. Each time a device wants to send data frames during the CAP, it locates the boundary of the next backoff slot and then waits for a random number of backoff slots. If the channel is busy, the device waits for another random number of backoff slots before attempting to access the channel again. If the channel is idle, it starts to send on the next available backoff slot boundary. Figure 2.4 shows the slotted CSMA/CA mechanism. For more details of slotted CSMA/CA operation, see [1].

In nonbeacon-enabled PANs, a device waits for a random period. If the channel is idle, it sends its data frame. If the channel is busy, it waits for another random period before

attempting to access the channel again.

The unslotted CSMA/CA channel access mechanism is similar to the Distributed Coordination Function (DCF) in the IEEE 802.11 [11]. However, it is simpler than the DCF as shown in Figure 2.5. When a WSD has a packet to transmit, it initializes the Number of Backoff (NB) and Backoff Exponent (BE) by 0 and $macMinBE$ respectively. Then, it waits for a randomly chosen backoff period, which belongs to $[0, 2^{BE} - 1]$. When the backoff timer is expired, the Clear Channel Assessment (CCA) for CSMA/CA is performed. By means of CCA, the WSD is able to see if the channel is busy. If the channel is sensed idle, the packet is transmitted. On the other hand, if the channel is busy, the WSD backoffs again for next transmission attempt, leading to increasing both NB and BE. If NB reaches $macMaxCSMABackoffs$, the packet will be dropped.

Some properties make a difference between the IEEE 802.15.4 MAC protocol and IEEE 802.11 DCF. Firstly, a DCF-enabled wireless device must wait for a period of time called DCF Interframe Space (DIFS) regardless of the packet length, while in the IEEE 802.15.4, the WSD waits for a period of either Long Interframe Spacing (LIFS) or Short Interframe Spacing (SIFS) depending on the packet length. Secondly, the IEEE 802.11 DCF pauses the backoff timer whenever the channel is sensed busy, while the IEEE 802.15.4 backoff mechanism does not stop the backoff timer even if other devices are using the channel. Thirdly, on a successful transmission, the DCF-enabled wireless device must wait for a period of SIFS to send an ACK, while the WSD does not hesitate to send the ACK. Lastly, the IEEE 802.11 defines a RTS/CTS hand-shake mechanism to alleviate the hidden terminal problem [12], while the IEEE 802.15.4 does not support it.

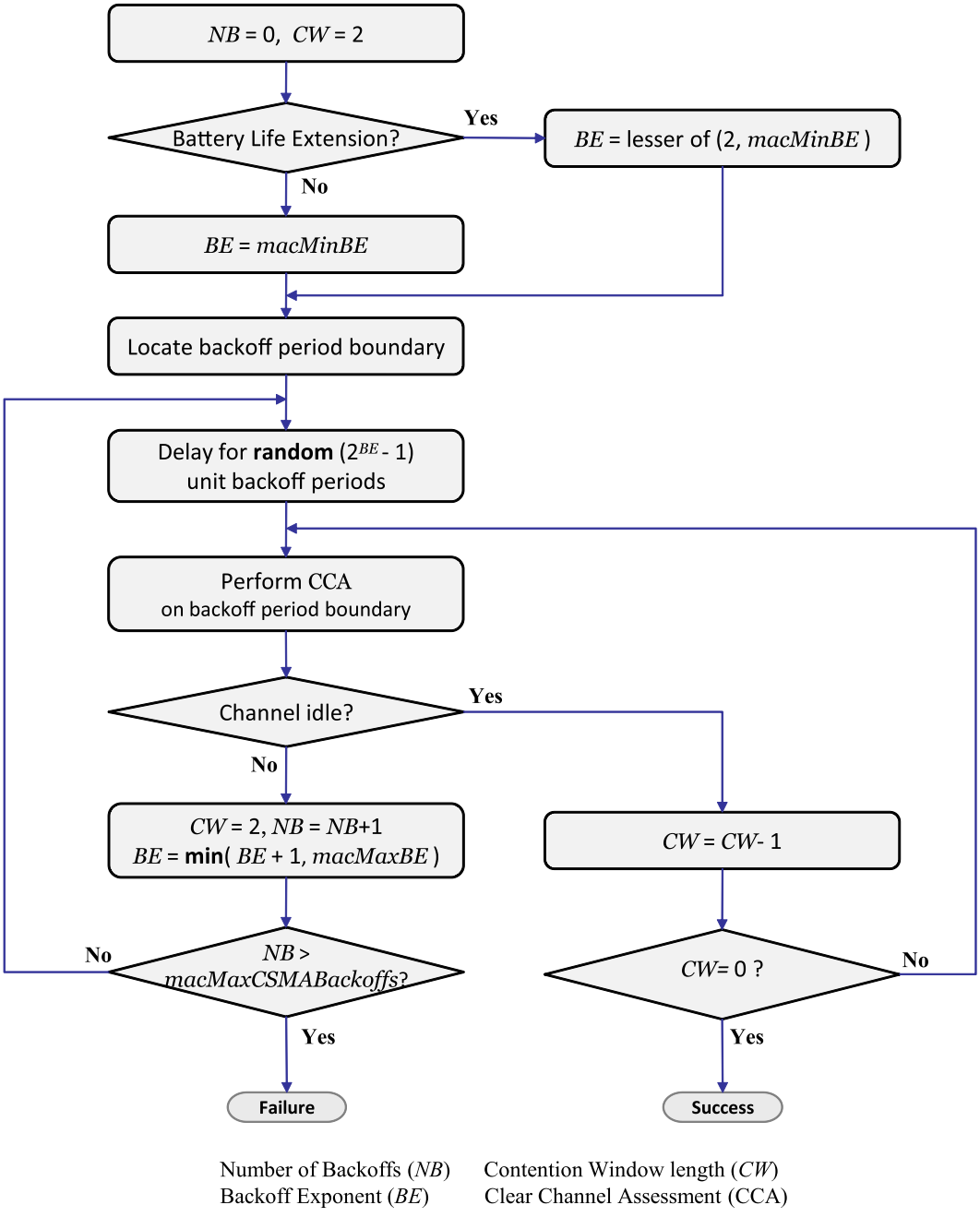


Figure 2.4: IEEE 802.15.4 slotted CSMA/CA channel access mechanism.

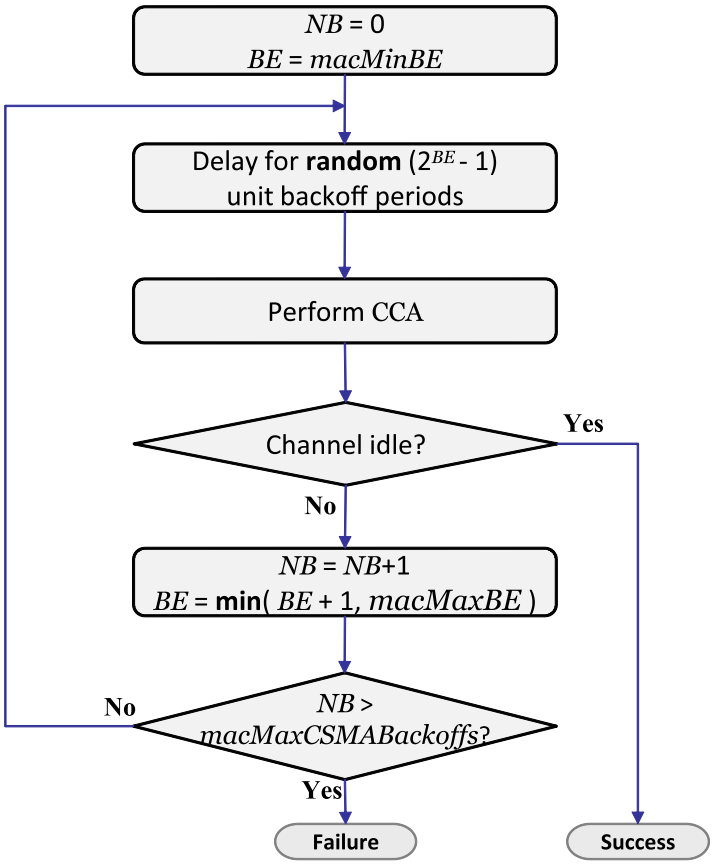


Figure 2.5: IEEE 802.15.4 unslotted CSMA/CA channel access mechanism.

2.5 Guaranteed Time Slot

The IEEE 802.15.4 MAC protocol defines Guaranteed Time Slot (GTS) for real-time service. The GTS starts at the end of CAP and is consecutively allocated until the end of SD. Seven out of sixteen time slots are reserved for such GTSs (See Figure 2.3).

The GTS reservation scheme is similar to Time Division Multiple Access (TDMA) scheme in terms of time slot reservation. However, For a TDMA, time is divided into frames of fixed duration and each frame is divided into a fixed number of time slots. These time slots are dedicated for the use of a connection so that some degree of bandwidth is guaranteed.

On the other hand, the GTS in the IEEE 802.15.4 is not fixed length, but adjustable by beacon parameters, i.e., BO and SO. Moreover, it supports for dynamic topology changes such as node failure and new nodes entering the network.

Basically, the IEEE 802.15.4 MAC protocol provides a wireless sensor with the capability to reserve GTSs at the coordinator, where wireless sensors can transmit one or more data packets in each of reserved GTSs. A wireless sensor requests GTSs from the coordinator, by sending a GTS request message.

The coordinator either accepts or rejects the request based on its admission control policy, which depends on the current resource capacity available in the superframe. GTSs are allocated on a FCFS basis, and must be placed at the end of the CAP. Once a GTS request from a wireless sensor is granted, the coordinator reserves the GTS for the wireless sensor during the CFP. Such a GTS mechanism is quite attractive for time-sensitive applications since it is possible to guarantee end-to-end packet delay bounds.

2.6 GTS Session Establishment and Closing

Figure 2.6 shows a message sequence diagram for GTS session establishment defined in the IEEE 802.15.4 standard. A wireless sensor sends a *GTSRequest* message to the coordinator that receives it and replies with an acknowledgment. If there are enough GTSs available for the wireless sensor, the beacon frame includes a GTS descriptor that has a starting index of the GTS and the number of GTSs allocated to the wireless sensor.

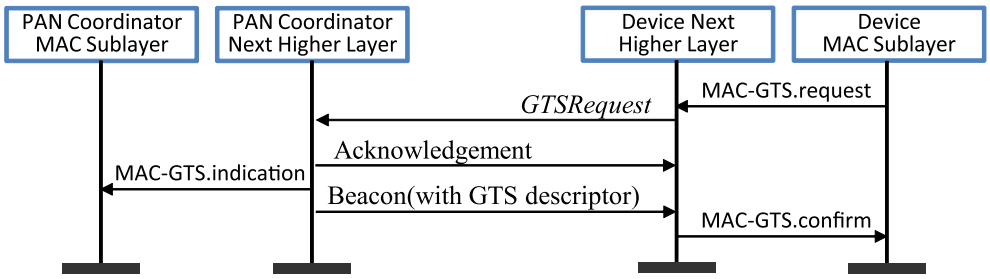


Figure 2.6: Message sequence diagram for GTS session establishment defined in the IEEE 802.15.4 [1].

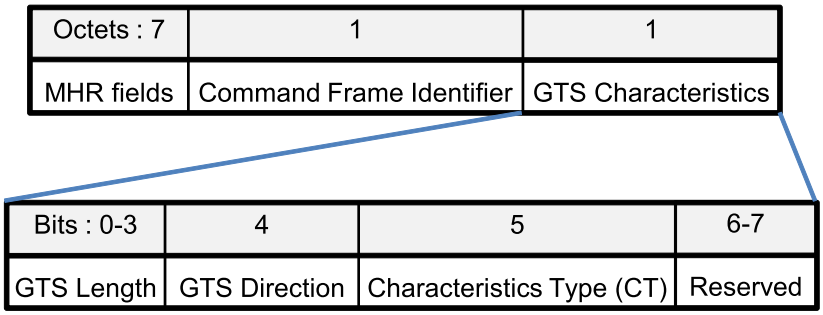


Figure 2.7: *GTSRequest* message format [1].

Two GTS session closing methods are defined. The first one is the explicit GTS session closing and the second one is the implicit GTS session closing.

For the first method, a wireless sensor closes its GTS session explicitly by sending a *GTSRequest* message with a characteristics type field of zero (See Figure 2.7).

For the second method, if there are no transmissions during the GTSs allocated to a wireless

sensor, the GTS session is closed by the coordinator. If a starting slot of the GTS equals zero, the GTS session is closed implicitly by the coordinator.

2.7 GTS Allocation and Scheduling Method

In the operating system context, scheduling is a method that chooses one of the processes (or tasks) to assign the CPU to them [13]. In the IEEE 802.15.4 context, on the other hand, scheduling methods select one of the transactions to allocate GTSs to these transactions. It is divided into a) FCFS [1], b) EDF [5] and c) priority-based methods [14].

As the name implies, the FCFS-based GTS scheduling method allows the coordinator to choose one of the transactions on a FCFS basis (See Figure 2.8(a)). In the EDF-based GTS scheduling method, the coordinator selects a transactions with the earliest delay constraint (See Figure 2.8(b)). In the priority-based GTS scheduling method, the coordinator selects a transaction with the highest priority (See Figure 2.8(c)).

Figure 2.8 shows an example of these GTS scheduling methods. We assume in the Figure that wireless sensor i corresponds to T_j, i.e., $i = j$.

Table 2.1: A set of transactions and parameters.

Node i	# of GTSs requested	Arrival time (a_j)	Priority (p_j)	Transaction (T_j)
1	1	a_1	p_1	$T_1(d_1, P_1)$
2	2	a_2	p_2	$T_2(d_2, P_2)$
3	2	a_3	p_3	$T_3(d_3, P_3)$
4	3	a_4	p_4	$T_4(d_4, P_4)$

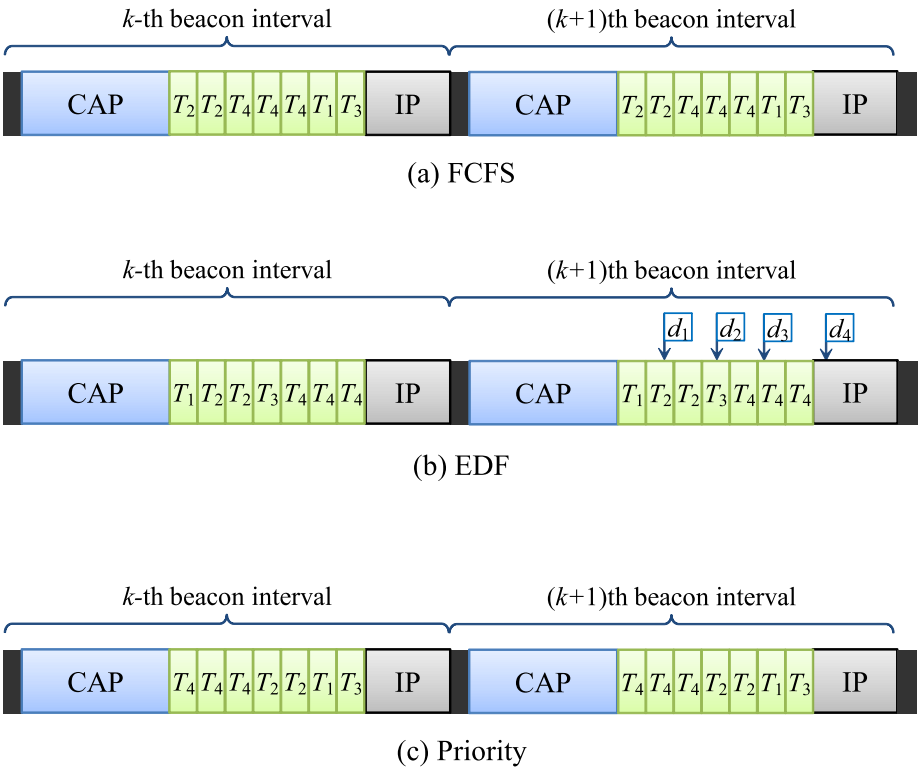


Figure 2.8: Three GTS scheduling methods with regard to Table 2.1. Note that $d_1 < d_2 < d_3 < d_4$, $a_2 < a_4 < a_1 < a_3$ and $p_4 < p_2 < p_1 < p_3$.

CHAPTER 3

OPTIMAL GTS ALLOCATION AND SCHEDULING

3.1 Introduction

In this chapter, we present a new GTS Allocation and Scheduling (GAS) algorithm to meet the delay constraints of delay-sensitive transactions in the beacon-enabled star network. Different from the EDF scheduling method [15], which results in bursty transmissions of payloads for transactions with delay constraints, it smooths out the traffic of a transaction by distributing the GTSs of the transaction over as many beacon intervals as possible while satisfying the delay constraint of the transaction.

By doing so, it reduces the average service starting time of transactions and enables concurrent services to more transactions. This can significantly benefit many delay-sensitive applications, where the starting time of the first message and the stability of traffic have great impact on the performance of these applications. Simulation results also demonstrate that the GAS significantly outperforms the FCFS-based GTS allocation and scheduling method defined in the IEEE 802.15.4 LR-WPAN. Our performance evaluation shows that the delay constraint meet ratio of our algorithm is up to 100% higher than the standard method. Our algorithm differs from the existing ones in that it is an on-line scheduling algorithm and allows transmissions of bursty and periodic messages with delay constraints even when the network is overloaded.

3.1.1 Existing Approaches

In wireless sensor networks, intensive research works have been done to satisfy various types of service requirements. Exiting works in [16] [17] [18] [19] [20] [21] proposed time slot assignment and scheduling mechanisms to meet delay requirements or to improve the time slot utilization.

In [17] [18], the authors presented an implicit GTS allocation mechanism in order to overcome a waste of time slots led by explicit GTS allocation. Since wireless sensors share the GTSs with each other, this algorithm improves the bandwidth utilization based on the delay and bandwidth requirements. Since their approach uses the FIFO-based GTS scheduling method defined in the IEEE 802.15.4 and provides no feasibility examination method, however, it fails to meet the delay requirements of real-time traffic and leads to GTS scarcity. Furthermore, it requires additional control packets that specify a flow status in the higher layer.

The authors in [16] presented a Time Division Multiple Access (TDMA) slot assignment protocol transmitted in a distributed manner. Its goal is to maximize slot utilization and to meet delay constraints by changing the frame length. If there are no available time slots, their proposed method makes sending nodes change the frame length such that the length of time slots is reduced and thus the number of available time slots increases. However, such adjusted length of time slots may incur longer delay. Another drawback is that it needs exchanging additional control packets.

In [19], the authors presented a GTS allocation method that divides a GTS into the small length of multiple slots to enhance the GTS utilization and to meet delay constraint. They use the fixed length of beacon interval and superframe duration such that GTSs are under-utilized if the length of packets does not fit the length of the GTSs. Further,as the approach in [16], their approach leads to longer packet delays as well as more energy consumption in that the packets must be fragmented to comply with a small length of time slots.

In [20], the authors presented a time-based scheduling algorithm that assigns time slots and variable-length guard times to satisfy the delay requirements. However, such guard times cause a longer delay so that the algorithm fails to provide delay guarantees.

The research work in [21] proposed the maximal traffic scheduling algorithm that allocates the same time slot to the transmitter-receiver pairs that do not cause interference. To find these pairs, the authors utilize a graph coloring technique and location information provided by UWB MAC, thus guaranteeing the maximum traffic and minimum time slots.

Some other research works in [22] [23] [24] have focused on adjusting superframe duration and beacon intervals to satisfy various types of traffic patterns.

The research work in [22] focused on adaptation of beacon parameters to meet the delay requirements of periodic messages. This approach takes only messages into account so that it fails to meet the delay requirements of bursty and aperiodic messages.

The research work in [23] uses the prioritized beacon exponent and toning schemes in a star topology. Basically, this approach assigns a wireless sensor with urgent messages a constant beacon exponent that is less than the beacon exponents of the other wireless sensors with normal messages so that it gives the urgent messages higher priorities. In particular, it avoids heavy collisions during the contention access period while it cannot handle the bursty transmissions of urgent messages with delay constraints. Further, since the IEEE 802.15.4 does not have built-in support for priorities, significant changes to the standard are required to use the proposed approach.

In [22], the authors proposed an off-line algorithm that adjusts both the superframe duration and the beacon interval depending on periodic transmissions of messages. This algorithm does not handle bursty arrivals of packets from wireless sensors and does not consider a per-transaction delay constraint such that it provides no delay guarantee for real-time traffic.

The authors in [24] presented an approach to adapt the beacon interval based on the recorded

communication frequency. In this approach, the length of beacon intervals increases or decreases depending on the frequency of communication among wireless sensors. The performance of the algorithm is highly sensitive to the values of the lower and upper bounds that determine the length of beacon intervals.

Some other research works in [25] [26] proposed scheduling methods for satisfying either delay constraints or periodic messages.

The research work in [25] proposed a channel scheduling algorithm that meets the delay requirements and allows for concurrent communication between Ultra Wideband (UWB) [27] transmitter-receiver pairs within a piconet.

The authors in [26] proposed a multi-cycle polling scheduling algorithm that yields a multi-cycle schedule for periodic messages in the FieldBus networks. Their polling algorithm leads to an overhead for the context switching since a master device periodically polls each slave device. Furthermore, before starting polling, it must determine a sequence of transmission and assignment of priorities to the messages. In addition, it underutilizes the time slots if there exist time spaces between the messages.

3.1.2 Chapter Organization

The remainder of this chapter is organized as follows. In Section 3.2, we address the problems of existing GTS allocation and scheduling methods, and give examples. In Section 3.3, we present a GTS session management including necessary components for establishing a new GTS session and give a brief description of the design of the GAS algorithm. In Section 3.4, we present how to examine the feasibility of transactions to make sure that all of them meet their delay constraints. In Section 3.5, we not only design an analytic mode that describes the delay of a transaction, but also present how to allocate the number of GTSs to the transactions. The description of the GAS is discussed in Section 3.6. In Section 3.7 we discuss

how to implement the GAS in the IEEE 802.15.4-enabled application. The performance evaluation, including the experimental results, is given in Section 3.8. Finally, we give a summary of the chapter in Section 3.9.

3.2 Problem Statement

The IEEE 802.15.4 standard specifies two types of channel access mechanisms. One is beacon-enabled mechanism and the other is nonbeacon-enabled one. The nonbeacon-enabled channel access mechanism aims at providing fair access to all wireless sensors and does not support delay-sensitive applications while the beacon-enabled channel access mechanism provides real-time guaranteed service by allocating the GTS on a FCFS basis, where GTS is a period of time that is reserved by a wireless sensor for real-time transmissions. Such applications commonly seen in medical or manufacturing sensor networks that monitor and signal emergencies.

In the beacon-enabled network all wireless sensors are synchronized by the beacon periodically transmitted by the coordinator. While the beacon-enabled network is preferred by delay-sensitive wireless applications, the FCFS-based real-time service defined in the IEEE 802.15.4 LR-WPAN has the following drawbacks.

- No delay guarantee

  It provides no delay guarantee scheme that examines whether the actual delay of a transaction is bounded by its delay constraint.

- GTS scarcity

  Since the GTS is limited resources, a wireless sensor's GTS request can be rejected if it requests more GTSs than what the coordinator can allocate.

- GTS underutilization

  Since the coordinator allocates as many GTSs as wireless sensors request, they can be underutilized or wasted if some portions of them are not used or they are allocated more than what the wireless sensors actually use.

- No overload control

  It does not provide any scheme that deals with a set of infeasible transactions where some of them miss their delay constraints. Without overload control, the coordinator cannot ensure that all the transactions satisfy their delay constraints.

Due to these critical drawbacks of the real-time service defined in the IEEE 802.15.4 standard, it is necessary to design a new algorithm that not only dynamically allocates GTSs, but also meets the delay requirements for those delay-sensitive transactions.

In this section we give examples of these drawbacks and show how they have the critical impact on delay guarantees. Figure 3.1 shows how the coordinator allocates GTSs to wireless sensors and schedules these GTSs in accordance with the IEEE 802.15.4 standard, i.e., FCFS-based GTS scheduling and static GTS allocation. For simplicity, we assume in the Figure that wireless sensor i corresponds to T_j, i.e., $i = j$. T_3 has the earliest delay constraint while T_1 has the latest delay constraint. On the other hand, T_1 has the earliest arrive time while T_3 has the latest arrival time.

It is observed that only T_1's delay constraint is guaranteed while the others miss their delay constraints. This is because the coordinator not only allocates GTSs with no consideration of the number of necessary GTSs, but also schedules them with no consideration of those transactions' delay constraints.

Figure 3.2 shows an example of GTS scarcity and GTS underutilization caused by the IEEE 802.15.4 standard. As in Figure 3.1, T_3 has the earliest delay constraint and the latest arrival time while T_1 has the latest delay constraint and the earliest arrival time. Let us assume that

Table 3.1: A set of transactions and parameters.

Node i	# of GTSs requested	Arrival time (a_j)	Transaction (T_j)
1	1	a_1	$T_1(d_1, P_1)$
2	4	a_2	$T_2(d_2, P_2)$
3	2	a_3	$T_3(d_3, P_3)$

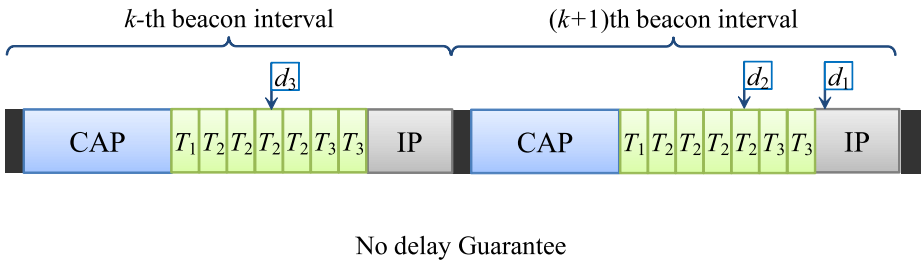


Figure 3.1: An example of delay guarantees performed by the IEEE 802.15.4 standard with respect to Table 3.1. Note that $d_1 > d_2 > d_3$ and $a_1 < a_2 < a_3$.

wireless sensors 1 and 3 request three and four GTSs respectively while the actual number of GTSs needed for them is two and three. We observe that wireless sensor 3's GTS request is rejected since the coordinator has not enough GTSs to service the wireless sensor even though there are two unused GTSs by wireless sensors 1 and 2 in the kth and $(k+1)$th beacon interval.

Table 3.2: A set of transactions and parameters.

Node i	# of GTSs requested	Arrival time (a_j)	Transaction (T_j)
1	3	a_1	$T_1(d_1, P_1)$
2	4	a_2	$T_2(d_2, P_2)$
3	2	a_3	$T_3(d_3, P_3)$

Furthermore, wireless sensors 1 and 2 request three and four GTSs respectively. However, wireless sensors 1 and 2 use two and three GTSs respectively such that two GTSs are not

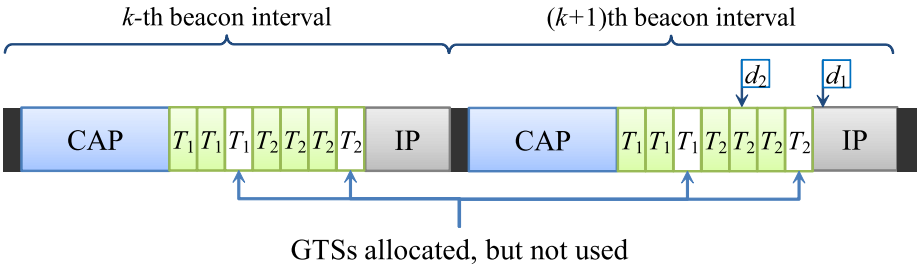


Figure 3.2: An example of GTS scarcity and GTS underutilization caused by the IEEE 802.15.4 standard with respect to Table 3.2. Note that $d_1 > d_2 > d_3$ and $a_1 < a_2 < a_3$.

used and underutilized in each beacon interval. This is because the IEEE 802.15.4 standard allocates GTSs statically without consideration of the actual number of GTSs needed by the wireless sensors.

In summary, the IEEE 802.15.4 standard method provides no delay guarantee for delay-sensitive transactions and incur the GTS scarcity and the GTS underutilization. To support such transactions with delay constraints and prevent such GTS scarcity and underutilization, it is necessary to design a new algorithm that not only dynamically allocates GTSs, but also meets the delay requirements of these transactions.

3.3 Design of the GTS Session Management for Delay-sensitive Transactions

In this section, we explain what components are required to maintain the GTS session as well as how to establish a new GTS session for a delay-sensitive transaction with a delay constraint.

3.3.1 Network Model and Assumptions

For the network model, we consider the beacon-enabled start topology (See Section 2.2). To begin with, we denote a set of wireless sensors as $\mathcal{N}$. A wireless sensor i is deployed within

the transmission range of the coordinator. All wireless sensors are synchronized with the coordinator by receiving beacon frames periodically and their transmissions are aligned with slot boundaries during the CFP. Further assumptions are as follows.

- AS1. BO and SO are fixed, which implies that both the BI and the SD are fixed.
- AS2. All wireless sensors are stationary.

It is worth noting that not only are the assumptions reasonable, but no strong assumptions are made.

3.3.2 Necessary Components for GTS Session Management

We define a transaction as a payload with a delay constraint that a wireless sensor i sends to the coordinator. We denote a set of transactions as $\mathcal{T}$, which is represented as $\mathcal{T} = \{T_j | 1 \leq j \leq m\}$ and is maintained by the coordinator until a transaction T_j is completed. Each transaction has a pair of (d_j, P_j, p_j), where d_j is the delay constraint in seconds, P_j is the payload length in bytes and p_j is the priority. All transactions in $\mathcal{T}$ are ordered in increasing delay constraints, i.e., the Earliest Delay Constraint First (EDCF) order.

Basically, two components are required to maintain a GTS session for delay-sensitive transactions. The first one is the admission control component, which plays a role in accepting or rejecting GTS requests from wireless sensors. For admission control, the feasibility examination is performed to ensure that a set of transactions meet their delay constraints, i.e., they are $feasible$. If some transactions are overloaded (which means some of them do not meet their delay constraints), it chooses one of the transactions in $\mathcal{T}$ and discards it to ensure that these transactions are still $feasible$ and they satisfy their delay constraints. The second one is the GTS allocation and scheduling component, which selects one of the wireless sensors, which are allocated the GTSs.

3.3.3 GTS Session Establishment with Session Parameters

In the start topology, the coordinator maintains GTS sessions created by wireless sensor i. Figure 3.3 illustrates how a new GTS session is established in response to wireless sensor i's request and how GTSs are allocated to them.

Firstly, a wireless sensor i sends a *GTSRequest* message enclosing a pair of a session parameter, (d_i, P_i)[1] to the coordinator (Step 1). Then, it replies with a *GTSAck*[2] message (Step 2). After that, the coordinator creates a new GTS session and a new transaction T_j with a transaction parameter, (d_j, P_j) and adds it to $\mathcal{T}$. If $\mathcal{T}$ is still $feasible$ after including the new transaction, the coordinator allocates and schedules the GTSs depending on the GTS allocation and scheduling methods. If the periodic beacon contains a GTS descriptor for wireless sensor i, then the wireless sensor sends data packets during the CFP (Step 3). Note that index j of a transaction is not necessarily equivalent to index i of a wireless sensor.

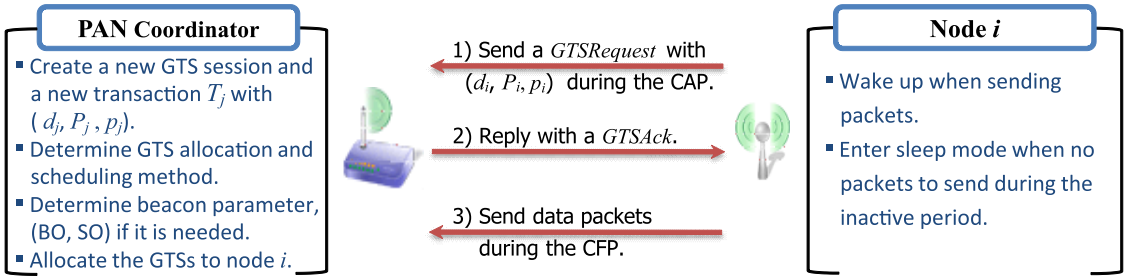


Figure 3.3: Message sequence diagram for GTS session establishment with session parameters.

3.3.4 Admission Control

Admission control regulates access to resources to avoid resource overload and to protect resources from requests that they cannot fulfill. It is based on some knowledge of both the

[1]To encapsulate it into a *GTSRequest* message, the MAC command frame defined in the standard is modified.

[2]It is equivalent to the acknowledgment frame in the standard.

overall GTS capacity and the load.

If a wireless sensor requests a large number of GTSs that the coordinator cannot afford to serve, its request is rejected implicitly. Namely, if a wireless sensor is not assigned any GTS until its delay constraint is expired, then its GTS request is rejected or its GTS session is closed.

This implicit rejection scheme does not require additional control messages to reject a GTS request or close a GTS session. As another mechanism, a wireless sensor can cancel its GTS session explicitly by sending a *GTSRequest* message with a characteristics type field of zero (See Figure 2.7).

It is possible that some transactions miss their delay constraints if such constraints are too short to complete the transactions. Such an overload condition may incur the *Domino Effect* [5], which causes existing feasible transactions to miss their delay constraints. Figure 3.4 shows such a phenomenon.

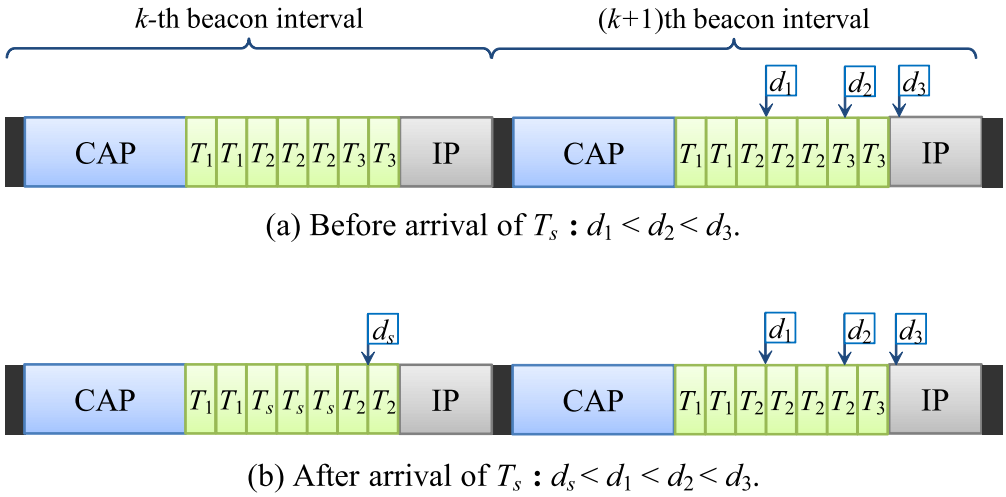


Figure 3.4: Domino effect on arrival of a new transaction T_s.

In Figure 3.4(a), T_1, T_2 and T_3 are feasible, which means that these transactions meet their delay constraints d_1, d_2 and d_3. In Figure 3.4(b), however, a new transaction T_s arrives at the ith beacon interval and is scheduled prior to T_2 and T_3 since its delay constraint d_s is earlier

than d_2 and d_3. As a result, T_s's arrival causes T_2 and T_3 to miss their delay constraints.

To prevent this phenomenon, a overload control scheme is required. One possible way is to discard a transaction that causes such a phenomenon. If the transaction causing the Domino Effect is the most important and urgent one out of the others, however, such a discard strategy is not acceptable in emergent places like nuclear power plants.

Another method involves a priority-based discard strategy that eliminates a transaction with the lowest priority.

3.3.5 Design of GTS Allocation and Scheduling Algorithm

In our design of GAS, we assume that beacon-enabled mode is used and a set of $|\mathcal{N}|$ wireless sensors with delay-sensitive data are deployed within the radio coverage of the coordinator. We also assume that a wireless sensor always transmits the largest amount of payload defined in the IEEE 802.15.4 standard subject to the constraint of the length of a GTS and that at least one GTS can be allocated to a wireless sensor.

When a wireless sensor i needs to transmit data packets, it sends a $GTSRequest$ message notifying the coordinator of its P_i and d_i. If the coordinator has enough GTSs to serve the request, it creates a new GTS session and assigns GTSs to the wireless sensor (See Section 3.3.3). The assignment is based on the consideration of two requirements. One is how many GTSs are needed by the payload P_i and the other is how to arrange these GTSs to satisfy the delay constraint d_i.

To satisfy these requirements, GAS is described in three phases. The first phase is the feasibility examination method making sure that all of the transactions are $feasible$, which means that they meet their delay constraints. For this method, we present both how to examine feasibility of these transactions and how to discard transactions to make sure that they remain feasible.

The second phase is the GTS allocation and scheduling method, called Earliest Delay Constraint with Minimum Guaranteed Time Slot (EDCF-mGTS), that assigns the minimum number of GTSs to as many transactions as possible in every beacon interval so that more transactions are served in every beacon interval, as compared with EDF. Namely, it is designed to support delay-sensitive applications where arrival of the first packet has a critical impact on the performance. For this, we design the delay model describing the total delay achieved by each transactions. Based on this delay model and feasibility examination method, we present how to reduce GTS scarcity and GTS underutilization.

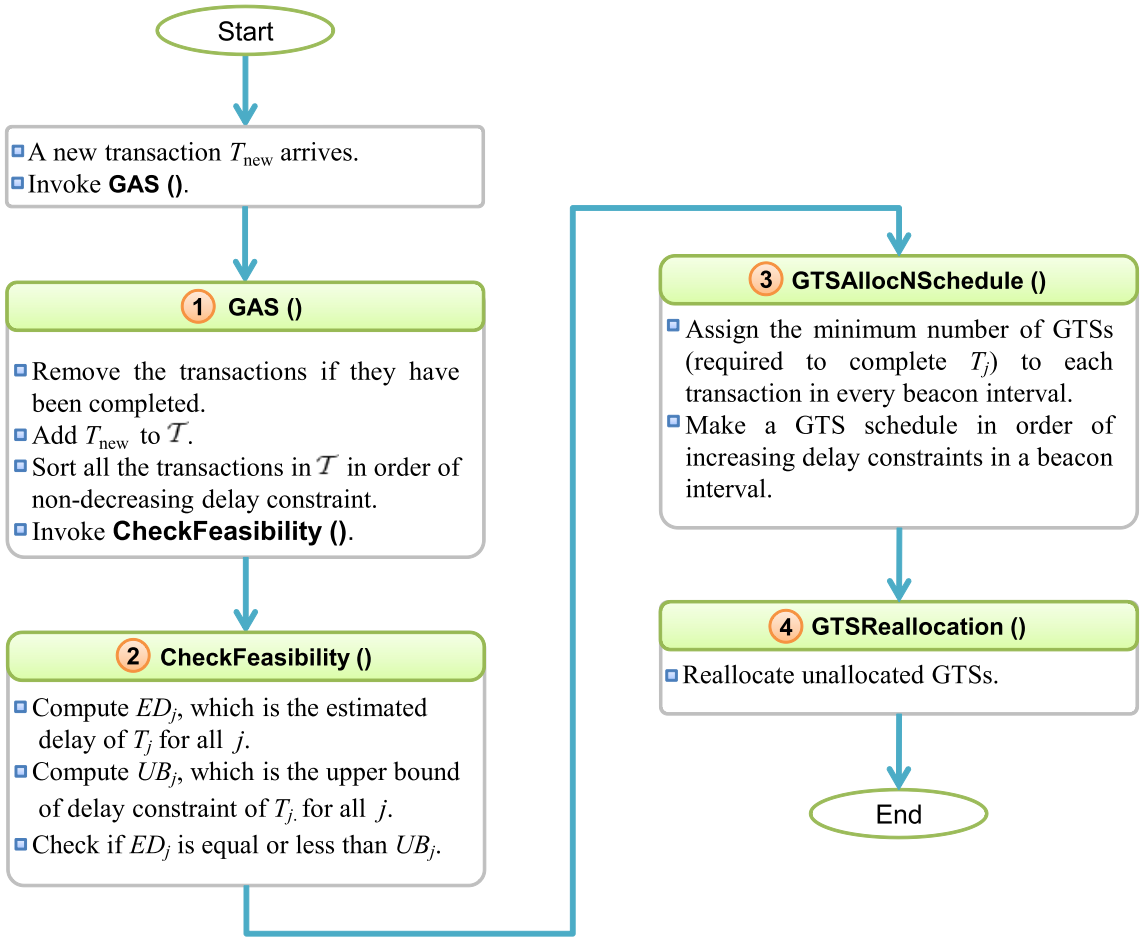


Figure 3.5: Control flow of the GAS.

The third phase is the GTS reallocation method that reassigns the GTS unallocated in a beacon interval. This method enables transactions to completed earlier than before GTS

reallocation so that it gives longer contention access periods for best-effort traffic.

Based on these three phases, in Section 3.5.4 GAS allocates the minimum number of GTSs to each transaction in each beacon interval so that $\mathcal{T}$'s payload can be maximally spread out.

To ensure that all GTSs are maximally utilized and the scheduling of GTSs is optimal, GAS adjusts the allocations of GTSs whenever the payload that needs to be transmitted in a CFP changes. These GTSs are evenly spread out over multiple beacon intervals to ensure a smooth traffic flow between the coordinator and wireless sensors. In the following sections, we discuss the details of this three-phase design.

3.4 Feasibility Examination

Before GAS starts to allocate GTSs to $\mathcal{T}$, it needs to check if $\mathcal{T}$ is feasible, which means that all the transactions in $\mathcal{T}$ meet their delay constraints. The feasibility examination has two steps. In the first step, GAS converts the delay constraint d_j of a transaction T_j into an adjusted relative delay constraint d_j^+ that reflects the real delay constraint for scheduling GTSs. Then, GAS examines whether all the transaction in $\mathcal{T}$ meet their delay constraints.

3.4.1 Adjustment of Delay Constraints

Given a transaction T_j, we first convert the absolute delay constraint[3] d_j to a relative delay constraint[4] d_j^{rdc} as:

$$d_j^{\text{rdc}} = d_j - t_{\text{WC}}, \tag{3.1}$$

[3]It is called the absolute delay constraint if a delay constraint is specified with respect to time zero.

[4]It is called the relative delay constraint if a delay constraint is specified with respect to the arrival time. In this report, we assume that the arrival time is the initial BI in which the first packet in T_j are sent.

where t_{WC} is the wall clock time. However, d_j^{rdc} does not reflect the real delay constraint that a transaction needs to meet. When a wireless sensor requests GTSs for its delay-sensitive transaction, d_j^{rdc} can be longer than the duration of a CFP in many cases. Since the coordinator can only allocate GTSs inside the CFP of a superframe, we need to convert the relative delay constraint d_j^{rdc} into an adjusted relative delay constraint, denoted as d_j^{+}, that is located inside the CFP. This d_j^{+} is the real delay constraint that the scheduling algorithm should meet.

For $d_j^{\mathrm{rdc}} \leq fCAP$, obvious, the transaction T_j is not feasible. For $d_j^{\mathrm{rdc}} > fCAP$, d_j^{+} can be expressed as:

$$d_j^{+} = \begin{cases} (\left\lceil \frac{d_j^{\mathrm{rdc}}}{BI} \right\rceil - 1)BI + CFP + fCAP, & \text{if } \kappa \geq fCAP + \mathrm{CFP}, \\ (\left\lceil \frac{d_j^{\mathrm{rdc}}}{BI} \right\rceil - 1)BI + \kappa, & \text{if } \kappa > fCAP \text{ and } \kappa < fCAP + \mathrm{CFP}, \\ (\left\lceil \frac{d_j^{\mathrm{rdc}}}{BI} \right\rceil - 2)BI + CFP + fCAP, & \text{if } \kappa \leq fCAP \text{ and } d_j^{\mathrm{rdc}} \geq BI, \end{cases} \quad (3.2)$$

where $fCAP$ is a duration from the first slot through the last slot of CAP and $\kappa = \mathbf{mod}(d_j^{\mathrm{rdc}}, BI)$. The three lines in Equation (3.2) can be explained by the three cases depicted in Figure 3.6. In Figure 3.6(a)'s case, d_1^{rdc} is located the inactive period of the $BI_h - th$ beacon interval, which means $\kappa \geq fCAP_h + CFP_h$.

By moving d_1^{rdc} from the inactive period into the closest CFP, CFP_h, we obtain d_1^{+} at the end of CFP_h, corresponding to the first line of Equation (3.2). In the case depicted in Figure 3.6(b), d_2^{rdc} is located in CFP_h, which implies $\kappa > fCAP_h$ and $\kappa < fCAP_h + CFP_h$. In this case, $d_2^{\mathrm{rdc}} = d_2^{+}$, corresponding to the second line of Equation (3.2).

In the case depicted in Figure 3.6(c), d_3^{rdc} is located in $fCAP_{h+1}$, which means that $\kappa \leq fCAP_{h+1}$ and $d_3^{\mathrm{rdc}} \geq BI_h$. In this case, the transaction with d_3^{rdc} should be completed right before the end of BI_h since the allocation of GTSs beyond BI_h cannot meet d_3^{rdc}. Thus, by moving d_3^{rdc} to the end of BI_h, we obtain d_3^{+}, which is expressed by the third line of Equation (3.2).

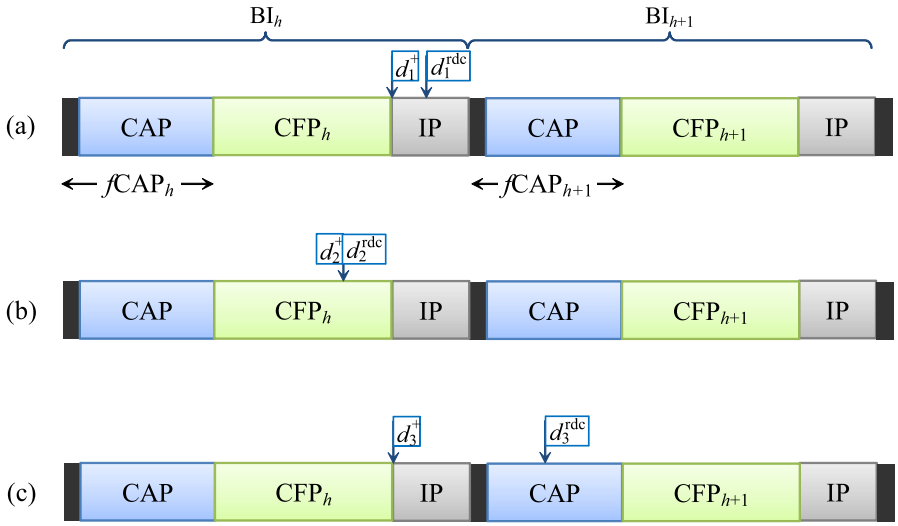


Figure 3.6: The adjustment of relative delay constraints.

Obviously, d_j^+ has the following properties.

Proposition 3.1 *Given a relative delay constraint $d_j^{rdc} > fCAP$ for a transaction T_j, its adjusted relative delay constraint d_j^+ calculated by Equation* (3.2) *is always inside a CFP and satisfies*

$$d_j^+ \leq d_j^{rdc}. \tag{3.3}$$

Proof. We consider two cases. In the first case, d_j^{rdc} is located beyond the CFP. By equation (3.2), d_j^{rdc} is placed in the closest CFP that meets d_j^{rdc}. Thus, $d_j^+ < d_j^{\text{rdc}}$. In the second case, d_j^{rdc} is located in the CFP. For this case, we do not need to adjust it to the closest CFP. Thus, $d_j^+ = d_j^{\text{rdc}}$. ■

The following corollary can be deduced from *Proposition* 3.1.

Corollary 3.1 *The adjusted relative delay constraint d_j^+ calculated by Equation* (3.2) *is always inside a CFP.*

3.4.2 Feasibility under Optimal Schedules

To examine the feasibility of existing transactions, we first calculate how many GTSs are required to complete T_j and the delay that these GTSs spend. Then, we examine whether such a delay is less than or equal to the upper bound of the delay.

Without loss of generality, assume that all transactions are sorted in order of increasing delay constraint so that $d_j^{\text{rdc}} < d_h^{\text{rdc}}$ for $j < h$. The number of GTSs that the coordinator allocates to a transaction is related to the amount of payload that can be transmitted in a GTS. To calculate this, note that the amount of time required for a wireless sensor to send a packet can be captured by a delay function as follows:

$$f_{\text{td}}(L_{\text{dpkt}}) = \frac{8 \times L_{\text{dpkt}}}{10^3 R}, \tag{3.4}$$

where L_{dpkt} is a data packet length and R is the data rate. Equation (3.4)[5] is the amount of time required to push a $8 \times L_{\text{dpkt}}$-bit packet into a wireless network. Denote by γ the duration taken for wireless sensor i to send a packet with the maximum packet length, $L_{\text{max-dpkt}}$. Based on Equation (3.4), γ can be expressed as:

$$\gamma = f_{\text{td}}(L_{\text{max-dpkt}}). \tag{3.5}$$

From Equation (3.5), the maximum number of packets transmitted by a wireless sensor during one GTS, denoted as $F_{\max}$, is:

$$F_{\max} = \left\lfloor \frac{\Delta t_{\text{gts}}}{\gamma + IFS} \right\rfloor + \left\lfloor \frac{\textbf{mod}(\Delta t_{\text{gts}}, \gamma + IFS)}{\gamma} \right\rfloor, \tag{3.6}$$

where IFS is the duration of an interframe space and Δt_{gts} is the duration of a GTS. We

[5]In the IEEE 802.15.4 beacon mode network, the propagation delay can be negligible since inter-node distance is very small [28].

assume that there is no acknowledgment for the reception of packets at the coordinator. Given $F_{\max}$, the number of GTSs needed to transmit packets in T_j is:

$$g_j = \left\lceil \frac{n_j}{F_{\max}} \right\rceil, \tag{3.7}$$

where n_j is the number of packets required to transmit payload P_j. Figure 3.7 shows an example of Δt_{gts}, γ and IFS.

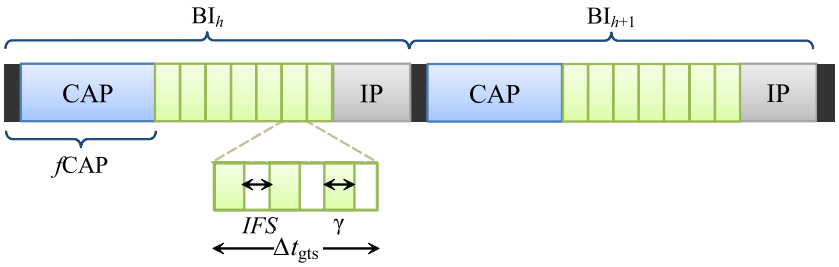


Figure 3.7: Δt_{gts}, γ and IFS. $F_{\max}$=3.

Based on Equation (3.7) and the fact that the EDF algorithm schedules transactions in order of increasing delay constraints, the EDF algorithm assign the first g_1 GTSs to T_1, the following g_2 GTSs to T_2, and so forth. It is obvious that the EDF results in bursty transmissions of the payload of a transaction. It also delays the service start time of transactions that have large delay constraints. Although the EDF has these undesirable features, it is an optimal and work-conserving algorithm. Therefore, it is good for a feasibility examination purpose.

With the above analysis about how EDF assigns GTSs to transactions, we can now proceed to check the feasibility of a set of transactions. Denote by β_j the maximum number of beacon intervals that a transaction T_j can use while still satisfying its adjusted relative delay constraint. β_j can be calculated as:

$$\beta_j = \left\lceil \frac{d_j^+}{BI} \right\rceil. \tag{3.8}$$

In these β_j BIs, the maximum amount of available time that can be used to serve transactions

before T_j's adjusted relative delay constraint d_j^+, is given by

$$UB_j = (\beta_j - 1)N_{\text{max-gts}}\Delta t_{\text{gts}} + \left[d_j^+ - (\beta_j - 1)BI - fCAP\right], \tag{3.9}$$

where $N_{\text{max-gts}}$ is the maximum number of GTSs that can be allocated in a CFP and is up to seven GTSs. The last component in Equation (3.9) is essentially the amount of time in the β_j-th beacon interval that can be used to serve transactions before the delay constraint d_j^+. The first component in Equation (3.9) is the amount of time that can be used to serve transactions in the previous $(\beta_j - 1)$ beacon intervals.

At the completion of transaction T_j, the actual amount of time spent in serving all the transactions, denoted as ED_j, can be calculated as

$$ED_j = \begin{cases} \widehat{ED}_j, & \text{if } j = 1 \\ \widehat{ED}_j + \Delta t_{\text{gts}} \sum_{k=1}^{j-1} g_k, & \text{if } 1 < j \leq |\mathcal{T}|, \end{cases} \tag{3.10}$$

where $\widehat{ED}_j$ is the aggregated amount of time serving transaction T_j in all the CFPs. Combining Equations (3.4) and (3.6), ED_j is:

$$\widehat{ED}_j = \begin{cases} f_{\text{td}}(L_{j,n_j}) + (\gamma + IFS)(n_j - 1), & \text{if } n_j \leq F_{\max} \\ f_{\text{td}}(L_{j,n_j}) + (\gamma + IFS)\left(\mathbf{mod}(n_j, F_{\max}) - 1\right) + \Delta t_{\text{gts}} \left\lfloor \dfrac{n_j}{F_{\max}} \right\rfloor, & \text{otherwise,} \end{cases} \tag{3.11}$$

where L_{j,n_j} is the length of the last packet of T_j.

Obviously, if EDF can schedule a transaction T_j to meet its delay constraint d_j^+, we must have

$$ED_j \leq UB_j \tag{3.12}$$

Hence, the feasibility condition of transactions can be expressed by Theorem 3.1.

Theorem 3.1 *Assume that transactions are sorted in order of increasing delay constraint so that* $d_j^{rdc} < d_k^{rdc}$ *for* $j < k$. *A set of transactions* $\mathcal{T}$ *is feasible under an optimal scheduling algorithm if and only if*

$$ED_j \leq UB_j, \ (\forall T_j \in \mathcal{T}). \tag{3.13}$$

Proof. (*If*) The EDF algorithm is an optimal scheduling algorithm. Therefore, if a set of transactions is feasible under the EDF algorithm, than the set of transactions is feasible under any optimal scheduling algorithm. When Equation (3.13) is true for any i, EDF can meet the delay constraint for all transactions. Hence, we prove the "if" condition.

(*Only if*) When Equation (3.13) is not true for some i, the completion time of T_j under EDF is larger than d_j^+. Hence, T_j is not feasible under EDF, which means that T_j is not feasible under any optimal scheduling algorithm. ■

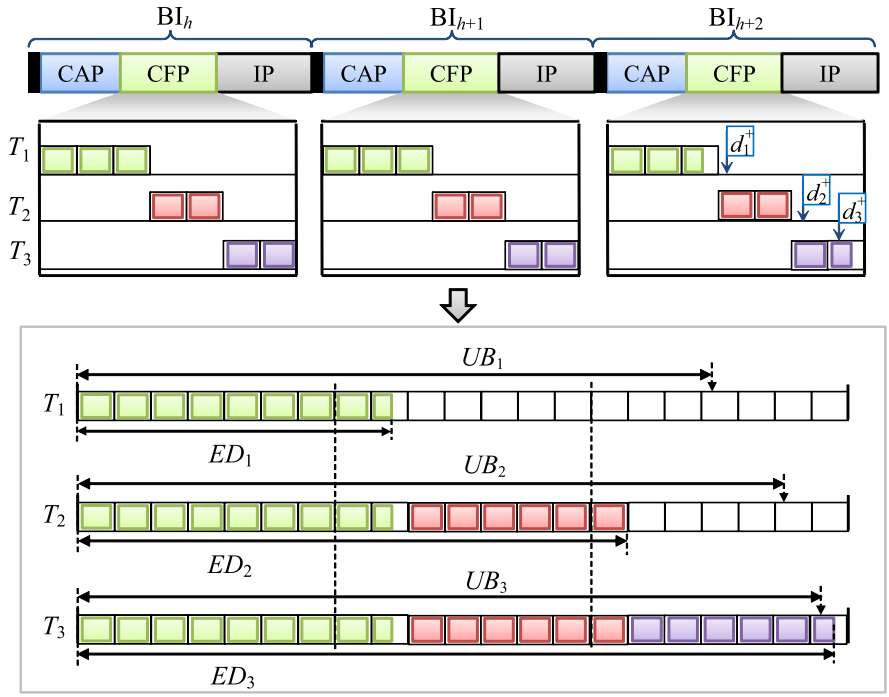


Figure 3.8: An example of feasibility examination for T_1, T_2 andT_3.

Figure 3.8 shows the feasibility examination for three transactions. Both ED_1^{cml} and ED_2^{cml} for T_1 and T_2 are less than UB_1 and UB_2, which indicates that the schedule consisting of a set of GTSs allocated to T_1 and T_2 is feasible. In case of T_3, however, $ED_3^{\text{cml}} > UB_3$, which means that the cumulative end-to-end delay caused by three transactions exceeds d_3^+. In this case, the schedule consisting of a set of GTSs allocated to three transactions is not feasible. To ensure that $\mathcal{T}$ is still feasible, T_j with the lowest priority will be discarded.

3.5 Cumulative Delay Model

In this section, we design the Delay Model for Fixed Beacon interval and Superframe duration (DM-FBS), which describes an expected delay between the beginning of the first beacon interval after T_j's arrival and the completion time of T_j. We denote it as D_j. It consists of two components.

Assuming transaction T_j is completed in b_j beacon intervals after its arrival, the first component is T_j's delay caused by the first $(b_j - 1)$ beacon intervals, which can be easily calculated as $(b_j - 1)BI$.

The second component is T_j's delay in the last (the b_j-th) beacon interval, which is related to the delay in transmitting T_j's data packets and waiting for other transactions' transmissions. To calculate this second component of D_j, in the remainder of this section, we first calculate the time required for T_j to send its data packets in the b_j-th beacon interval.

Then, we present how to calculate the number of GTSs scheduled prior to T_j's GTSs in the b_j-th beacon interval. Finally, by summing up both components of D_j, we derive D_j.

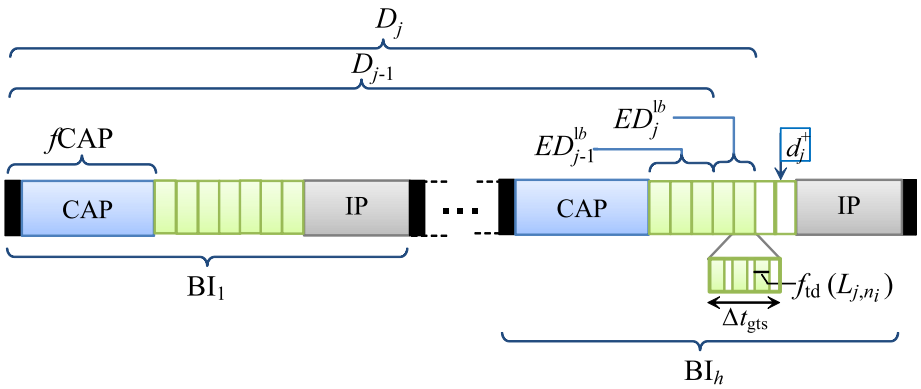


Figure 3.9: An example of D_j and ED_j^{lb}: $F_{\max} = 3$ and $s_j = \widetilde{s}_j = 2$ in T_j's b_j-th beacon interval, i.e., BI_h.

3.5.1 Calculating the Delay of T_j in T_j's last Beacon Interval

In this section, we calculate the time for sending transaction T_j's own packets in its last beacon interval. To do this, we first need to figure out how many T_j's data packets are to be sent in its last beacon interval. Given that s_j GTSs are allocated to a transaction T_j in each beacon interval, T_j can be completed in b_j beacon intervals where

$$b_j = \left\lceil \frac{n_j}{s_j \times F_{\max}} \right\rceil . \tag{3.14}$$

where s_j is the number of GTSs allocated to T_j in each beacon interval (See Section 3.5.4). The number of packets in T_j transmitted in the b_j-th beacon interval is calculated as:

$$\widetilde{f}_j = \mathbf{mod}(n_j, s_j \times F_{\max}), \tag{3.15}$$

Since $\widetilde{f}_j$ may not fill s_j GTSs in the b_j-th beacon interval, the number of GTSs allocated to T_j in the b_j-th beacon interval is given by:

$$\widetilde{s}_j = \begin{cases} \left\lceil \dfrac{\widetilde{f}_j}{F_{\max}} \right\rceil, & \text{if } \widetilde{f}_j \neq 0 \\ s_j, & \text{otherwise.} \end{cases} \tag{3.16}$$

Based on Equations (3.6), (3.15) and (3.16), the delay that it takes to transmit $\widetilde{f}_j$ in the b_j-th beacon interval of T_j is given by

$$ED_j^{lb} = (\widetilde{s}_j - 1)\,(\Delta t_{\text{gts}} - F_{\max}(\gamma + IFS)) + (\gamma + IFS)(\widetilde{f}_j - 1) + f_{\text{td}}(L_{j,n_j}), \tag{3.17}$$

where L_{j,n_j} is the length of the last packet of T_j. Figure 3.9 illustrates an example of ED_j^{lb} for $F_{\max}$=4 and $s_j = \widetilde{s}_j = 2$.

Proposition 3.2 *For a transaction T_j, the following inequality holds.*

$$\widetilde{s}_j \leq s_j. \tag{3.18}$$

Proof. We consider two cases.

Case 1: For $\mathbf{mod}(n_j, s_j \times F_{\max}) = 0$, since $\widetilde{f}_j = s_j \times F_{\max}$, the number of GTSs in the b_j-th beacon interval is s_j. Hence, $\widetilde{s}_j = s_j$.

Case 2: For $\mathbf{mod}(n_j, s_j \times F_{\max}) \neq 0$, since $\widetilde{f}_j < s_j \times F_{\max}$, the following inequality holds.

$$\frac{\widetilde{f}_j}{F_{\max}} < s_j. \tag{3.19}$$

By taking the ceiling of the left side, we obtain

$$\left\lceil \frac{\widetilde{f}_j}{F_{\max}} \right\rceil \leq s_j. \tag{3.20}$$

Since the left side of inequality (3.20) is the number of GTSs allocated to T_j in the b_j-th beacon interval,

$$\widetilde{s}_i = \left\lceil \frac{\widetilde{f}_i}{F_{\max}} \right\rceil . \tag{3.21}$$

Hence, combining Inequality (3.20) and equality (3.21) gives $\widetilde{s}_i \leq s_j$. ■

3.5.2 Calculating the number of GTSs allocated before T_j in T_j's last Beacon Interval

To calculate the delay in serving transaction T_j, we need to find out how many transactions are scheduled prior to T_j in the b_j-th beacon interval since the number of GTSs allocated to these transactions affects the length of time that T_j must wait before it is completed in the b_j-th beacon interval.

In each beacon interval, the GTSs are scheduled in order of increasing delay constraints of the transactions, essentially applying the EDF algorithm inside each beacon interval. To obtain the number of GTSs scheduled before T_j and to calculate the duration that T_j must wait in the b_j-th beacon interval, we introduce the Dependent Slot List (DSL) defined as follows.

Definition 3.1 *For a transaction T_j, DSL is defined as*

$$DSL = \left\{ e_{T_j}[b, s, \widetilde{s}, d^{rdc}] \ : \ 1 \leq j \leq |\mathcal{T}| \right\} . \tag{3.22}$$

The following *Proposition* 3.3, *Proposition* 3.4, *Proposition* 3.5 explain how to use DSL to calculate the number of GTSs allocated before T_j in the b_j-th beacon interval.

Proposition 3.3 *For any transaction T_k, $(k < j)$, scheduled before T_j in the b_j-th beacon interval, e_{T_k} must precede e_{T_j} in DSL.*

Proof. For any transaction T_k, $k < j$, that succeeds T_j in DSL, either $e_{T_k}[b] < e_{T_j}[b]$ or $e_{T_k}[b] = e_{T_j}[b]$ and $e_{T_k}[d^{\text{rdc}}] > e_{T_j}[d^{\text{rdc}}]$ hold.

For the first case where $e_{T_k}[b] < e_{T_j}[b]$, in the $e_{T_j}[b]$-th beacon interval, transaction T_k has no GTS to schedule since T_k has already been completed before the $e_{T_j}[b]$-th beacon interval.

For the second case where $e_{T_k}[b] = e_{T_j}[b]$ and $e_{T_k}[d^{\text{rdc}}] > e_{T_j}[d^{\text{rdc}}]$, since GAS schedules the GTSs of transactions in order of increasing delay constraints of these transactions, T_k's GTSs are scheduled after T_j's GTSs. This proves the proposition. ■

Proposition 3.4 *For any e_{T_k} that precedes e_{T_j}, $(k \neq j)$, in DSL, a transaction T_k is scheduled before a transaction T_j in the b_j-th beacon interval if and only if $e_{T_k}[d^{rdc}] < e_{T_j}[d^{rdc}]$.*

Proof. (**If**) Since e_{T_k} precedes e_{T_j} $(k \neq j)$, in DSL, either $e_{T_k}[b] > e_{T_j}[b]$ or $e_{T_k}[b] = e_{T_j}[b]$ and $e_{T_k}[d^{\text{rdc}}] < e_{T_j}[d^{\text{rdc}}]$ hold. In both cases, T_k has GTSs to be scheduled in the $e_{T_j}[b]$-th beacon interval.

In the first case where $e_{T_k}[b] > e_{T_j}[b]$, if $e_{T_k}[d^{\text{rdc}}] < e_{T_j}[d^{\text{rdc}}]$, T_k's GTSs are scheduled before T_j's GTSs since GAS schedules the GTSs of transactions in order of increasing delay constraints of these transactions.

In the second case where $e_{T_k}[b] = e_{T_j}[b]$ and $e_{T_k}[d^{\text{rdc}}] < e_{T_j}[d^{\text{rdc}}]$, T_k's GTSs are scheduled before T_j's GTSs since GAS schedules GTSs in order of increasing delay constraints.

(**Only if**) If $e_{T_k}[d^{\text{rdc}}] > e_{T_j}[d^{\text{rdc}}]$, T_k's GTSs are scheduled after T_j's GTSs since GAS schedules GTSs of transactions in order of increasing delay constraints of these transactions.

This proves the proposition. ■

Proposition 3.5 *For any transaction T_k, $(k \neq j)$, that schedules its GTSs before T_j in the b_j-th beacon interval, if $e_{T_k}[b] = e_{T_j}[b]$, then T_k schedules $e_{T_k}[\widetilde{s}]$ GTSs before T_j. If $e_{T_k}[b] > e_{T_j}[b]$, then T_k schedules $e_{T_k}[s]$ GTSs before T_j.*

Proof. For the first case where $e_{T_j}[b] = e_{T_k}[b]$, the $e_{T_j}[b]$-th beacon interval of T_j is equivalent to the last beacon interval of T_k. Hence, the $e_{T_k}[\widetilde{s}]$ GTSs allocated to transaction T_k are scheduled before T_j.

For the second case where $e_{T_k}[b] > e_{T_j}[b]$, since the $e_{T_j}[b]$-th beacon interval of T_j is not equivalent to the last beacon interval of T_k, the number of GTSs allocated to T_k in the $e_{T_j}[b]$-th beacon interval equals $e_{T_k}[s]$ and these $e_{T_k}[s]$ GTSs are scheduled before T_j. This proves the proposition. ■

Algorithm 1 describes the pseudo code that builds a DSL. `BuildDSL()` adds a new DSL element to the DSL in order of decreasing b_j. If $e_{T_j}[b]$ is equivalent to $e_{T_k}[b]$ for $j \neq k$, they are ordered by increasing $e_{T_j}[d^{\mathrm{rdc}}]$.

After building the DSL, we look it up to find out the number of GTSs allocated before T_j in the b_j-th beacon interval. To do this, we examine which transactions are placed prior to T_j in the b_j-th beacon interval.

Algorithm 1: BuildDSL()

```
1  Input  : e_{T_new}[b, s, s̃, d^rdc] ∉ DSL
2  Output: None
3  Data: k
4  for k ← 1 to |DSL| do
5      if e_{T_new}[b] > e_{T_k}[b] then
6          Link e_{T_new} to the head of e_{T_k}.
7          return
8      end
9      else if e_{T_new}[b] = e_{T_k}[b] AND e_{T_new}[d^rdc] < e_{T_k}[d^rdc] then
10         Link e_{T_new} to the head of e_{T_k}.
11         return
12     end
13 end
14 Link e_{T_new} to the tail of DSL.
```

Algorithm 2: fdsl()

Input : $e_{T_j}[b, s, \widetilde{s}, d^{\text{rdc}}] \in$ DSL

Output: λ_{slot} , wihch is the number of GTSs scheduled before T_j in the $e_{T_j}[b]$-th beacon interval

Data: k, λ_{slot}

$k \leftarrow 1$

$\lambda_{\text{slot}} \leftarrow 0$

while $e_{T_k}[d^{rdc}] \neq e_{T_j}[d^{rdc}]$ **do**

  if $e_{T_k}[d^{rdc}] > e_{T_j}[d^{rdc}]$ AND $e_{T_k}[b] < e_{T_j}[b]$ **then**

    $\lambda_{\text{slot}} \leftarrow \lambda_{\text{slot}} + e_{T_k}[s]$

  end

  else if $e_{T_k}[d^{rdc}] > e_{T_j}[d^{rdc}]$ AND $e_{T_k}[b] = e_{T_j}[b]$ **then**

    $\lambda_{\text{slot}} \leftarrow \lambda_{\text{slot}} + e_{T_k}[\widetilde{s}]$

  end

  $k \leftarrow k + 1$

end

return λ_{slot}

Based on *Proposition* 3.3, *Proposition* 3.4 and *Proposition* 3.5, Algorithm 2 shows the pseudo code for calculating the number GTSs scheduled before T_j in the b_j-th beacon interval. Lines 7 to 12 implement *Proposition* 3.5.

To better illustrate Algorithm 2, consider the input parameter is $e_{T_3}[2, 2, 2, d_3^{\text{rdc}}]$. $e_{T_2}[3, 2, 1, d_2^{\text{rdc}}]$ increases λ_{slot} to $e_{T_2}[s]$ by the condition in line 7 of Algorithm 2. $e_{T_4}[3, 1, 1, d_4^{\text{rdc}}]$ does not increase λ_{slot} since $e_{T_3}[d_3^{\text{rdc}}] < e_{T_4}[d_4^{\text{rdc}}]$, which means that T_1 is scheduled prior to T_3 since T_1's delay constraint is earlier than T_3's delay constraint.

For $e_{T_1}[2, 2, 2, d_1^{\text{rdc}}]$, $e_{T_1}[\widetilde{s}]$ is added to λ_{slot} since $e_{T_3}[d_3^{\text{rdc}}] > e_{T_1}[d_1^{\text{rdc}}]$ and $e_{T_3}[b] = e_{T_1}[b]$ satisfy the condition in line 9. As a result, the number of GTSs scheduled prior to T_3 is the summation of both $e_{T_2}[s]$ and $e_{T_1}[\widetilde{s}]$, which equals 4. Figure 3.10 illustrates an example of a DSL built by these operations.

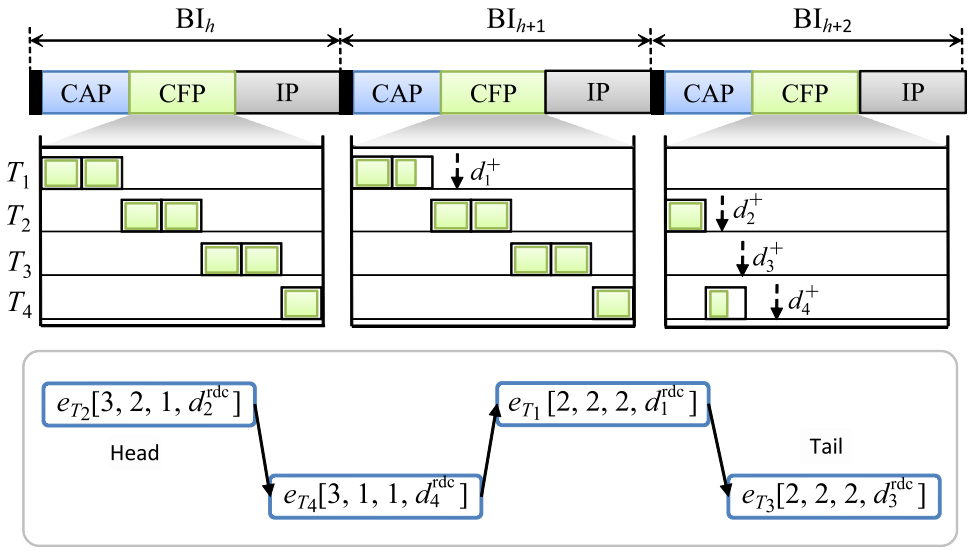


Figure 3.10: GTS allocation and DSL: $d_1^{\text{rdc}} = d_1^+ < d_2^{\text{rdc}} = d_2^+ < d_3^{\text{rdc}} = d_3^+ < d_4^{\text{rdc}} = d_4^+$.

The advantage of DSL is that when we need to find out the number of GTSs scheduled before T_j, with DSL, we just need to scan the transactions that are located before e_{T_j} in DSL. However, without DSL, we must look up all transactions to calculatet the number of GTSs scheduled before T_j.

3.5.3 Putting All together

As the final step of the mathematical modeling, we formulate D_j, the delay for transmitting all of a transaction T_j's packets. Combining Equation (3.17) and Algorithm 2, we derive D_j as follows:

$$D_j = (b_j - 1)BI + fCAP + \Delta t_{\text{gts}} \times \mathbf{fdsl}(j) + ED_j^{\text{lb}}. \tag{3.23}$$

3.5.4 Making a GTS schedule in each Beacon Interval

In this section, we present how to bound this delay to make sure that it is smaller than T_j's delay constraint d_j^+ by allocating the minimum number of GTSs to T_j in each beacon interval. Note that to ensure T_j meets its delay constraints, the estimated transmission delay must satisfy the following condition:

$$D_j \leq d_j^+, \tag{3.24}$$

where d_j^+ is given in Equation (3.2) and D_j is given in Equation (3.23).

Based on Equations (3.14), (3.17) and (3.23), the value of D_j depends on s_j, which is the number of GTSs allocated to T_j in each CFP.

Therefore, by calculating the minimum s_j that satisfies Inequality (3.24) using Equations (3.2), (3.14), (3.17) and (3.23), we get the minimum number of GTSs, denoted as $s_j^{\min}$, that must be allocated to a transaction to meet its delay constraint in each beacon interval.

It is worth noting that allocating $s_j^{\min}$ GTSs to transaction T_j may not be feasible since $s_j^{\min}$ is calculated by assuming that we must allocate the same number of GTSs to a transaction in all the beacon intervals except the last one. Therefore, due to rounding up the number of GTSs in each interval to an integer, some transactions are over-allocated with GTSs.

Since we allocate GTSs to transactions in order of increasing delay constraint, the over-allocation to transactions that have smaller delay constraints than T_j can cause a shortage of GTSs to T_j. Hence, the real minimum number of GTSs that can be allocated to T_j is

$$s_j = \min\{s_j^{\min}, N_{\text{max-gts}} - \sum_{k:d_k^{\text{rdc}}<d_j^{\text{rdc}}} s_k\}. \tag{3.25}$$

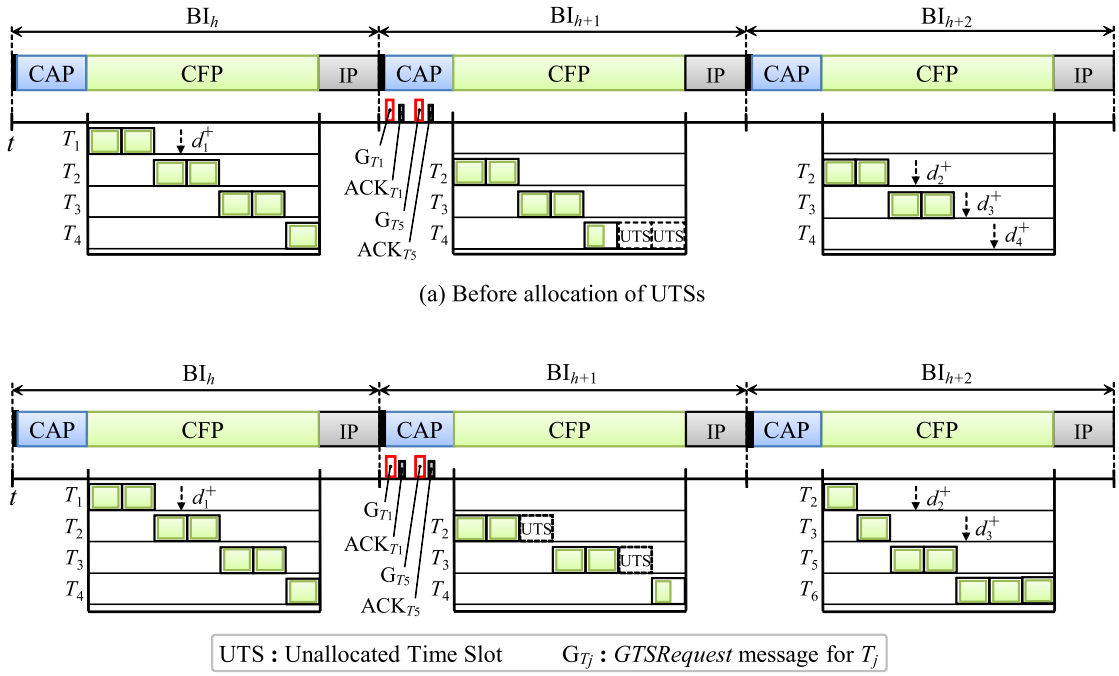
In every beacon interval, the minimum number of GTSs meeting d_j^+ is assigned to T_j in order of increasing delay constraints until this assignment reaches $N_{\text{max-gts}}$. We call this method Earliest Delay Constraint First with Minimum GTS (EDCF-mGTS) in that the minimum number of GTSs becomes large in every beacon interval if d_j^+ is short.

It may also be possible that by assigning $s_j^{\min}$ to each transaction, we still have some GTSs unallocated, called Unallocated GTS(UTS). Figure 3.11(a) shows an example of this case, where there are two UTSs in the BI_{h+1}.

To maximize the GTS utilization and ensure that GAS is work-conserving, we need to assign these UTSs to active transactions as depicted in Figure 3.11(b). A UTS is not necessarily allocable to any transaction. For example, in Figure 3.11(a), we should not assign both UTSs to T_4 since T_4 only needs one GTS to complete its payload in the BI_{h+1}. The maximum number of UTS that can be allocated to a transaction is determined by *Proposition* 3.6.

Proposition 3.6 *Given a set of transactions and their s_j, the number of UTSs in each beacon interval is $N_{UTS} = \max(0, N_{max\text{-}gts} - \sum_{T_j \in \mathcal{T}} s_j)$. If the total number of GTSs required for $T_j \in \mathcal{T}$, denoted as g_j in Equation (3.7), satisfies $g_j - s_j = x > 0$, then at most $\min(x, N_{UTS})$ number of UTSs can be allocated to transaction T_j.*

Based on *Proposition* 3.6, Algorithm 3 describes how GAS assigns UTSs to transactions.



(a) Before allocation of UTSs

(b) After allocation of UTSs

Figure 3.11: GTS Reallocation.

Algorithm 3: MaximizeGTSUtilization()

Input : $\mathcal{T}, \sigma = \{< T_j, s_j > | 1 \leq j \leq |\mathcal{T}|\}$
Output: None
Data: j, ρ_{slot}, g_j, `flag`
$\rho_{\text{slot}} \leftarrow \sum_{T_j \in \mathcal{T}} s_j$
if $N_{max\text{-}gts} - \rho_{slot} \leq 0$ **then return**
Compute g_j for $\forall j$, using Equation (3.7)
`flag` $\leftarrow$ `TRUE`
$j \leftarrow 1$
while `flag` = `TRUE` **do**
    if $g_j - s_j > 0$ **then**
        $s_j \leftarrow s_j + 1$
        $\rho_{\text{slot}} \leftarrow \rho_{\text{slot}} + 1$
    end
    if $\rho_{slot} = N_{max\text{-}gts}$ **then return**
    if $i = |\mathcal{T}|$ **then** $j \leftarrow 1$
    else $j \leftarrow j + 1$
end

The temporary shortage of GTSs for some transactions due to $s_j < s_j^{\min}$, however, does not pose a problem for the optimality of GAS, nor does it cause any violation of the delay constraint of T_j. This is because by over-allocating GTSs to transactions with smaller delay constraint than T_j, these transactions are completed earlier.

After these transactions are completed, they release the reserved GTSs and we can just recalculate the GTSs schedule to make sure that the lost GTSs can be regained by transaction T_j. The GTS reallocation process of GAS is invoked right before and after every beacon interval that has a transaction T_j completed in it or after the arrival of any new transactions.

When the reallocation process is invoked, for each transaction, GAS calculates the remaining payloads that yet to be served. Then, based on the remaining payloads, GAS reruns the GTS allocation algorithm described in Sections 3.5.1 to 3.5.4.

This reallocation algorithm ensures that the transactions that do not get enough GTSs in previous beacon intervals can obtain more GTSs in the following beacon intervals.

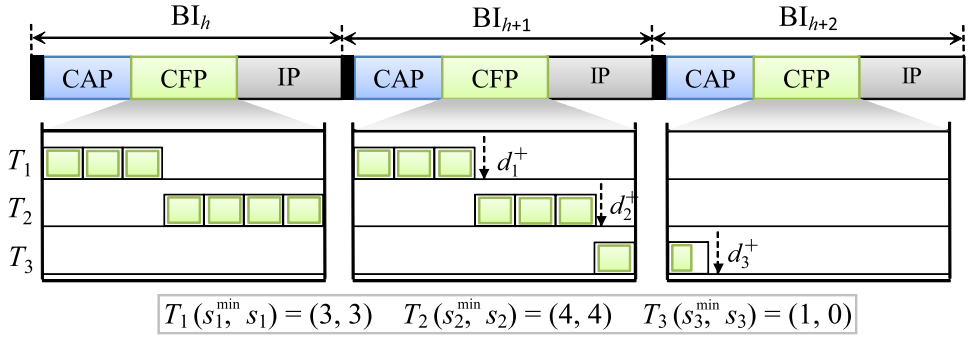


Figure 3.12: GTS Reallocation.

To better illustrate this reallocation process, consider an example depicted in Figure 3.12. T_1 and T_2 are assigned $s_1(=3)$ and $s_2(=4)$ respectively while T_3 is assigned $s_3(=0)$ by Equation (3.25).

This means that s_3 of T_3 cannot be allocated in the BI_{h+1} even though all these transactions are feasible by Inequality (3.13). In this case, s_3 can be allocated in the BI_{h+1} since T_1

and T_2 are completed in the BI_{h+1}. Therefore, GAS reallocates s_3 to T_3 at the BI_{h+1}-th scheduling event.

3.6 Summary of GAS Design

In this section, we summarize the design of GAS and prove its optimality. GAS is invoked before the coordinator transmits the beacon frame to the wireless sensors within its transmission range.

3.6.1 Algorithm Description

The GAS algorithm maintains a set of active transactions and is invoked at the beginning of each beacon interval. It first checks if a new GTS allocation needs to be calculated. In three cases, such recalculation is needed.

In the first case (lines 5 to 6 in Algorithm 4), some transactions will be completed in this beacon interval and hence will only use a smaller number of GTSs than in the previous beacon intervals. Hence, a reallocation of GTSs is needed to ensure these extra GTSs can be used by other transactions.

In the second case (lines 7 to 9 in Algorithm 4), some transactions were completed in the previous beacon interval and hence release their allocated GTSs. Hence, GAS needs to remove these transactions from Γ and calculate a new allocation of GTSs to ensure the released GTSs can be used by the remaining transactions.

In the final case (lines 10 to 15 in Algorithm 4), a new transaction arrives and GAS needs to check if this new transaction is feasible and calculate the GTSs allocation to the new

Algorithm 4: GTS Allocation and Scheduling Algorithm (GAS)

```
   /* Called at the beginning of each beacon interval                    */
1  Input  : T
2  Output: None
3  Data:  σ, needReallocation
4  needReallocation ← FALSE
5  if Some transactions will be completed in this beacon interval then
6  |  needReallocation ← TRUE
7  end
8  if Some transactions were completed in previous beacon interval then
9  |  Remove completed transactions from T
10 |  needReallocation ← TRUE
11 end
12 if Arrival of new transactions in the previous beacon interval then
13 |  foreach new transaction T_j do
14 |  |  Insert T_j to T
15 |  |  SortTrByEDCF(T)
16 |  |  if CheckFeasibility(T) = infeasible then Remove T_j from T
17 |  |  else needReallocation ← TRUE
18 |  end
19 end
20 if needReallocation = TRUE then
21 |  SortTrByEDCF(T)
22 |  σ ← GTSAllocNSchedule(T)
23 end
24 Allocate GTSs to transactions according to σ
```

transaction. Before calling `GTSAllocNSchedule()` and `CheckFeasibility()`, transactions need to be sorted in order of increasing delay constraint by calling `SortTrByEDCF()`.

Algorithm 5 allocates and reallocates the minimum number of GTSs to each active transaction. The input parameter to `GTSAllocNSchedule()` is $\mathcal{T}$, and the output is a schedule σ that describes the number of GTSs allocated to each $T_j \in \mathcal{T}$ in each beacon interval in the sequence of the actual schedule.

Algorithm 5: GTSAllocNSchedule()

```
1  Input  : T
2  Output: T
3  Data: s_alloc, s_j^min, j
4  Sort T_j ∈ T in order of increasing delay constraint.
5  s_alloc ← 0
6  for j ← 1 to |T| do
7  |   s_j^min ← 1
8  |   Compute D_j and increase s_j^min by 1 until D_j ≤ d_j.
9  |   s_alloc ← s_alloc + s_j^min
10 |   if s_alloc > N_max-gts then break
11 |   s_j ← s_j^min
12 end
13 if s_alloc < N_max-gts then GTSRealloc(T)
14 return T
```

Algorithm 6 examines whether a set of active transactions is feasible. The input parameter to `CheckFeasibility()` is $\mathcal{T}$ and the output is either feasible or infeasible. These two Algorithms are called by Algorithm 4.

Theorem 3.2 *For m transactions, the asymptotic complexity of GAS is $O(m^2 \log m)$.*

Algorithm 6: CheckFeasibility()

```
1  Input  : T
2  Output: Either feasible or infeasible
3  Data: λ_slot, j, g_j, ED_j, ED_j^cml, UB_j^cml
4  λ_slot ← 0
5  for j ← 1 to |T| do
6      Compute ED_j and g_j
7      if j = 1 then ED_j^cml ← ED_j
8      else
9          λ_slot ← λ_slot + Δt_gts g_{j-1}
10         ED_j^cml ← ED_j + λ_slot
11     end
12     Compute UB_j^cml
13     if UB_j^cml − ED_j^cml < 0 then return infeasible
14 end
15 return feasible
```

Proof. The asymptotic complexity of GAS depends on the procedures `GTSAllocNSchedule()`, `SortTrByEDCF()`, `CheckFeasibility()`, `GTSRealloc()` and the **for** loop in GAS. For m transactions, `GTSAllocNSchedule()` costs $O(m)$, `SortTrByEDCF()` costs $O(m \log m)$, `CheckFeasibility()` costs $O(m)$, and `GTSRealloc()` costs $O(m)$. The **for** loop iterates at most m times, thus resulting in $O(m^2 \log m) = O(m \log m) + O(m) \times O(m) + O(m \log m) + O(m)$. ■

Theorem 3.3 *If EDF can schedule a set of transactions and satisfy their delay constraints, GAS can also schedule this set of transactions and satisfy their delay constraints.*

Proof. Assume that we are given a set of transactions that are feasible under EDF and the transactions are sorted in the order of increasing delay constraints. From the design of GAS, it is easy to see that GAS always tries to allocate at least $s_j^{\min}$ GTSs to transactions.

Based on the calculation of $s_j^{\min}$, if GAS can allocate $s_j^{\min}$ GTSs to every transaction, the

delay constraints of all transactions are satisfied. However, due to over-allocation, GAS may run out of GTSs and only allocate $s_j < s_j^{\min}$ GTSs to some transactions.

Since GAS allocates GTSs to transactions in order of increasing delay constraint, the transactions that are allocated with enough GTSs ($s_j = s_j^{\min}$) have smaller delay constraints than the transactions that do not get $s_j^{\min}$ GTSs. Based on this observation, we can prove the optimality of GAS using an induction hypothesis.

Basis: It is easy to see that if under EDF, transaction T_1, which has the smallest delay constraint, is feasible, T_1 is also feasible under GAS.

Induction hypothesis: Assume that transaction $T_{k|k\leq j}$ are all feasible by EDF and GAS and transaction T_{j+1} is feasible under EDF. We now prove that transaction T_{j+1} is also feasible under GAS. If after allocating GTSs to T_j and all $T_{k|k<j}$, there are no less than s_{j+1}^{min} unallocated GTSs in each beacon interval, then GAS can allocate s_{j+1}^{min} GTSs to T_{j+1} and T_{j+1} is feasible.

If the number of unallocated GTSs in each beacon interval is smaller than s_{j+1}^{min}, GAS allocates all the remaining unallocated GTSs to T_{j+1} and all the transactions $T_{k|k>j+1}$ cannot get any GTS from GAS. Hence, none of the transactions that have larger delay constraints than T_{j+1} can affect T_{j+1}.

When a transaction $T_{i|i<j+1}$ is completed, GAS reallocates GTSs based on each transactions' remaining payloads. T_{j+1} either gets $s_j^{\min}$ GTSs in the reallocation and hence can meet its delay constraint, or it still gets $s_j < s_j^{\min}$ GTSs from GAS. In the later case, T_{j+1} waits for additional transactions with $T_{i|i<j+1}$ to be completed and no transaction $T_{k|k>j+1}$ can obtain GTSs.

The process continues and in the worst case, T_{j+1} has to wait until the last transaction among $T_{i|i<j+1}$ is completed. In this case, GAS has only allocated GTSs to transactions $T_{i|i<j+1}$ and T_{j+1}, where $\sum_{i=1}^{j} g_j$ GTSs are allocated to transactions $T_{i|i<j+1}$ and another group of GTSs

are allocated to T_{j+1}, denoted as x.

After the completion of the last transaction among $T_{i|i<j+1}$, GAS allocates GTSs to transaction T_{j+1}, which now requires $(\xi_{j+1} - x)$ GTSs to transmit its payload, and transactions $T_{k|k>j+1}$, each of which requires ξ_k GTSs.

The time and status of the transactions is exactly the same as the moment when EDF has completed transactions $T_{k|k<j+1}$ and has in total allocated x GTSs to transaction T_{j+1}. Note that T_{j+1} is the transaction with the smallest delay constraint among the remaining transactions.

According to the basis, the transaction with the smallest delay constraint is always feasible under GAS if it is feasible under EDF. Since EDF can schedule T_{j+1} to satisfy its delay constraint. Hence, T_{j+1} is feasible under GAS. We prove the induction hypothesis. ■

3.7 Implementation Considerations

From a practical point of view, the GAS can be easily integrated into the IEEE 802.15.4 protocol with a minor change. It has no protocol overhead since it does not need to exchange additional control packets, as compared with other protocols. At a wireless sensor, only transaction parameters are inserted into a *GTSRequest* message when a node has transactions to be transmitted.

Once the *GTSRequest* message reaches the coordinator during the CAP, the GAS puts it into the linearly-linked list, called GTS Transaction List (GTL). Then, it is invoked either right before sending a new BI or right after the CAP. In terms of resource consumption, it needs only two linearly linked lists, GTL and DSL whose the required amount of memory depends on the number of active wireless nodes around the coordinator.

Since the maximum number of GTSs are up to 7, however, such an amount of memory can be adjusted depending on the available resources that wireless sensor nodes can provide. Despite the small use of memory resources, the GAS involves two implementations (feasibility examination and GTS assignment) in consuming the energy. The GTS assignment consumes more energy than the feasibility examination since the GAS runs it every beacon interval while the feasibility examination is performed only when new $GTSRequest$ messages are received.

3.8 Performance Evaluation

In this section, we evaluate the performance of GAS in terms of its ability to satisfy the time constraints and its ability to achieve high GTS utilization using the ns-2 simulator [29].

3.8.1 Simulation Settings

For simulation, we consider the start topology with seven wireless sensors located in a 20×20 area. We set up transaction sets TS_k, where $1 \leq k \leq 30$, each of which consists of transactions $T_{i,j}$, where $1 \leq i \leq 7$ and $1 \leq j \leq 50$. The length of payload for transaction $T_{i,j}$ is selected uniformly from 1 kbyte to 150 kbytes. Each wireless sensor transmits 50 transactions per simulation run and the total number of runs is 30. Figures 3.13 and 3.14 show a geographical deployment of seven wireless sensors and an instance of transaction sets respectively.

Note that the priority for each transaction is randomly assigned. We compare GAS with another three algorithms. The first algorithm is FCFS-based GTS scheduling and allocation, which statically assigns as many GTSs as wireless sensor i request and schedules them on a FCFS basis. The second algorithm is EDF, which statically assigns as many GTSs as

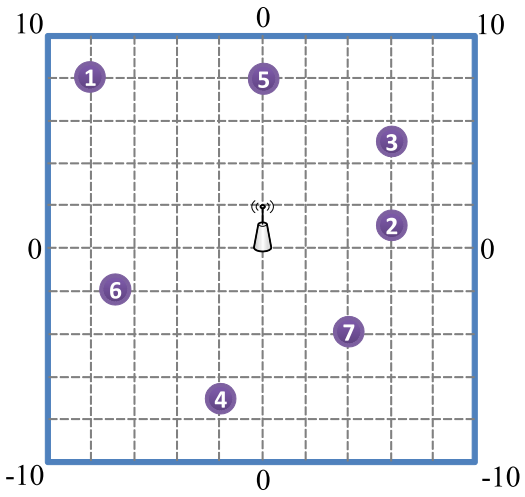


Figure 3.13: Geographical deployment of wireless sensors.

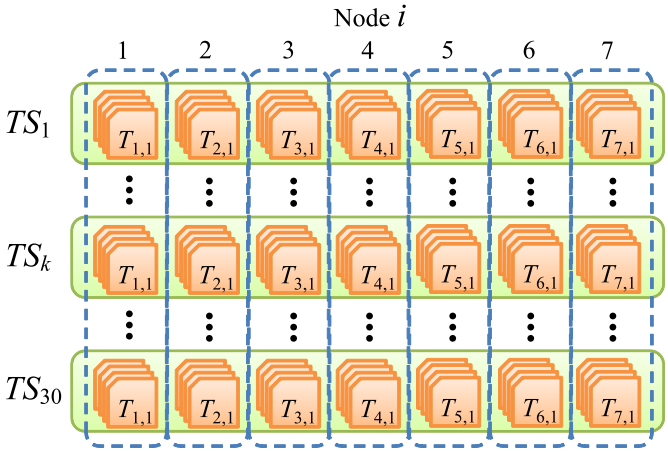


Figure 3.14: Transaction sets TS_k and transactions $T_{i,j}$, where $1 \leq k \leq 30$, $1 \leq i \leq 7$ and $1 \leq j \leq 50$.

wireless sensor i request and schedules them in order of increasing delay constraints.

Our evaluation considers three types of transmissions: bursty , periodic and aperiodic transmissions. We assume that there is no difference in the channel conditions of any wireless sensor and that only delay-sensitive transactions are transmitted in the network. Table 3.3 shows the 802.15.4 parameters for the simulation. All other parameters are the same as the defaults specified in the standard.

Table 3.3: 802.15.4 parameters for the simulation

Parameter	Value	Unit
aMaxFrameOverhead	9	Bytes
aMaxFrameOverhead	9	Bytes
aMaxPHYPacketSize	127	Bytes
aMaxMACFrameSize	118	Bytes
aMaxBeaconOverhead	75	Bytes
aMaxLIFSPeriod	40	Symbols
aMaxSIFSPeriod	12	Symbols
BeaconOrder	8	Symbols
SuperframeOrder	8	Symbols
BeaconInterval	245760	Symbols
SuperframeDuration	245760	Symbols
DataRate	250	kbps

3.8.2 Performance Metrics

For performance evaluation, we define the following performance metrics:

Delay constraint Meet Ratio (DMR) is the ratio of the number of transactions meeting their delay constraints to the number of transactions served, which is expressed as:

$$\text{DMR}(\%) = \frac{\text{The number of transactions meeting their delay constraints}}{\text{The number of transactions served}} \times 100. \quad (3.26)$$

It indicates how many transactions served by an algorithm meet their delay constraints.

Transaction Abort Ratio (TAR) is the ratio of the number of aborted transactions to the number of transactions requested by wireless sensors. It is expressed as:

$$\text{TAR}(\%) = \frac{\text{The number of aborted transactions}}{\text{The number of transactions requested}} \times 100. \quad (3.27)$$

It indicates how many transactions are rejected when the overloads condition or TCR > 1 occurs.

L_{max} is the maximum latency of the completion time of a transaction with respect to its delay constraint. It is given by

$$L_{max}(msec) = \max_i(x_{T_j} - d_{T_j}) \quad (3.28)$$

where x_{T_j} is the termination time of T_j and d_{T_j} is its absolute delay constraint that T_j must meet.

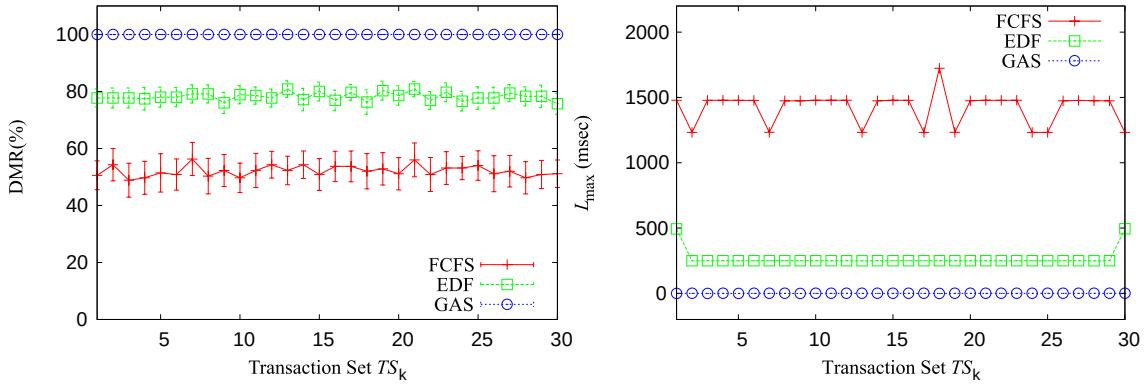
UG is the GTS utilization, which means how many GTSs are allocated to transactions over the beacon intervals, which is calculated as:

$$\text{UG}(\%) = \frac{\text{The total number of GTSs allocated}}{(\text{The number of beacon intervals transmitted} - 1) \times N_{\text{max-gts}}} \times 100. \quad (3.29)$$

Equation (3.29) excludes the first beacon interval since there are no transactions scheduled in the first CFP.

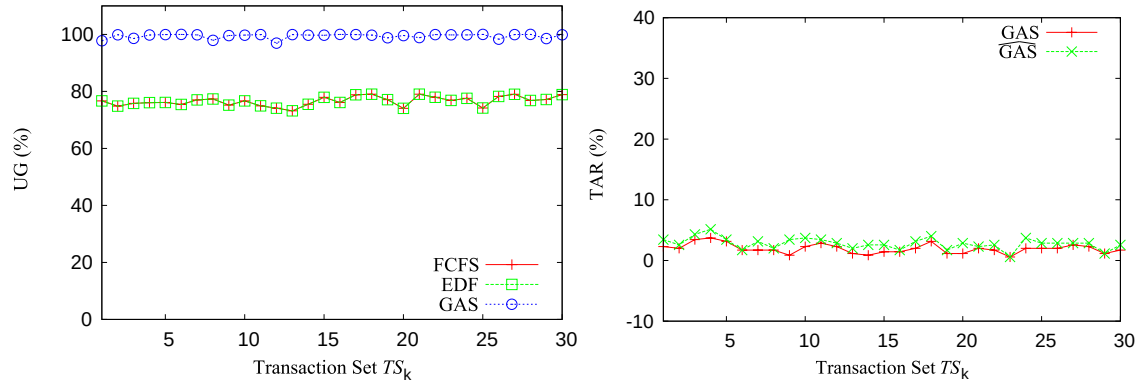
3.8.3 Bursty Transmissions of Transactions

In this section, we consider bursty transmissions of transactions, where the payload of a transaction arrives in a single burst.



(a) DMR with 99% confidence intervals

(b) L_{max}



(c) Utilization of GTS (UG)

(d) TAR

Figure 3.15: Performance under bursty transmissions.

Figures 3.15(a)-(c) show the DMR, L_{max} and UG of the FCFS, EDF and GAS under bursty transmissions. Each point is averaged over 50 of transactions. It can be observed that GAS achieves 100% of DMR and negative L_{max}, while both FCFS and EDF have positive L_{max} and significantly smaller DMRs than GAS due to their lack of feasibility examination. FCFS has lower DMR and larger L_{max} than EDF since it assigns GTSs to the transactions on a FCFS basis without considering their delay constraints. GAS maintains 100% DMR and negative L_{max} because it is optimal so that if a group of transactions are feasible under some

scheduling algorithm, GAS can always schedule them to satisfy their delay constraints.

Figure 3.15(c) shows that the UG of GAS is significantly higher than the other algorithms, demonstrating GAS's ability to maximize the utilization of GTSs. GAS's UG is slightly smaller than 100% for some transaction sets since in the last beacon interval where all transactions are completed, some UTSs may get left since no transactions need additional GTSs.

Figure 3.15(d) demonstrates the benefits of Algorithm 3 by comparing GAS with $\widehat{\text{GAS}}$, which is GAS without using Algorithm 3. Since Algorithm 3 ensures maximal usage of GTS, it shortens the completion time of transactions, which reduces TAR.

3.8.4 Periodic Transmissions of Transactions

In this section, we consider transactions whose payload arrives periodically. For this simulation, we set up transaction sets TS_k, where $1 \leq k \leq 30$, each of which consists of transactions, T_j, where $1 \leq j \leq 7$. Segments of a transaction T_j's payload arrive periodically for 50 intervals. The delay constraint of each arrived segment is the segment inter-arrival time. Table 3.4 shows an example of TS_1's settings.

Table 3.4: An example of TS_1 setting for periodic transmission.

T_j	P_j (bytes)	Period(msec)	d_j	# of GTSs requested
T_1	6095	2 × BI	6295	1
T_2	26411	6 × BI	19144	1
T_3	16817	5 × BI	11141	1
T_4	39797	8 × BI	27299	1
T_5	8005	3 × BI	6610	1
T_6	14282	4 × BI	10794	1
T_7	627	1 × BI	2237	1

For example, transaction T_1 has a 6095 Bytes payload segment arrives in every two beacon intervals. Hence, the 6095 Bytes has a delay constraint of the duration of two beacon intervals.

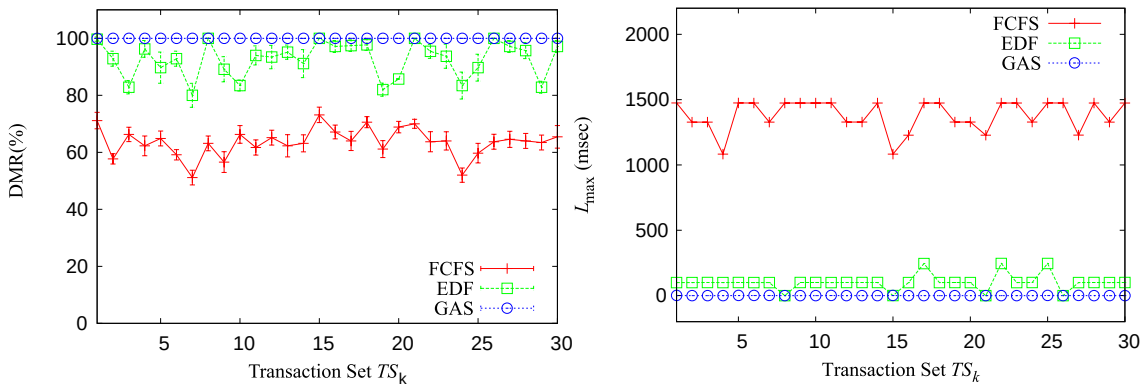
Figure 3.16(a)-(c) shows the DMRs, L_{max} and UG of FCFS, EDF and GAS under the periodic transmissions of transactions. Similar to the bursty arrival cases, GAS maintains 100% of DMR and negative L_{max} while FCFS and EDF have positive L_{max} and smaller DMRs than GAS. FCFS again has lower DMR and larger L_{max} than EDF due to its FCFS-based GTS scheduling method.

In Figure 3.16(c), the UG of GAS is comparable to the other algorithms. In some transaction sets, the UG of GAS is slightly smaller than that of FCFS and EDF. This is because GAS rejects transactions that are not feasible, which may potentially reduce utilization of GTSs as there may not be enough active transactions to use all GTSs. Comparing with Figure 3.15(d), Figure 3.16(d) shows that the benefits of Algorithm 3 is even larger for periodic transactions than for bursty transactions.

3.8.5 Aperiodic Transmissions of Transactions

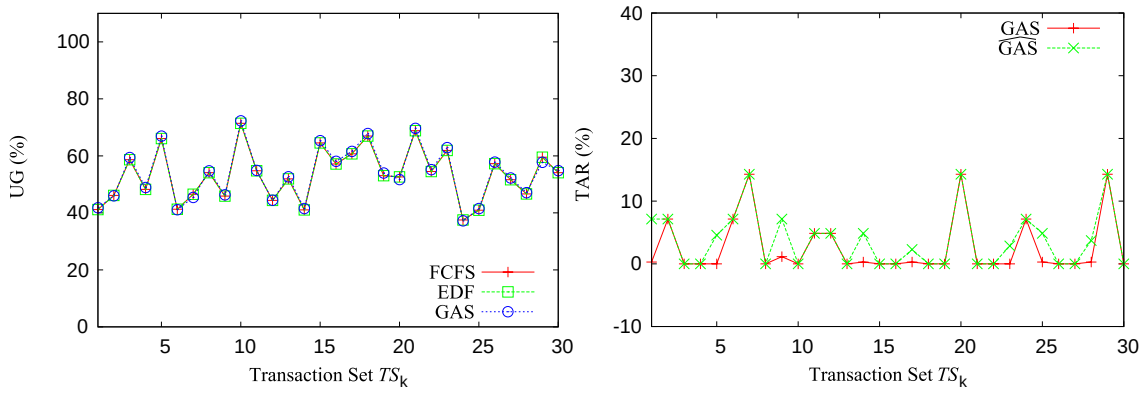
In this section, we consider aperiodic arrivals of transactions. We assume that a transaction $T_{i,j}$ from a wireless sensor i should be completed before the arrivals of its subsequent transaction $T_{i+1,j}$ at the same wireless sensor. The inter-arrival time between $T_{i,j}$ and $T_{i+1,j}$ is proportional to the payload size of $T_{i+1,j}$ with a random variation of 3 beacon intervals.

Figure 3.17(a)-(c) shows the DMRs, L_{max} and UG of FCFS, EDF and GAS under the aperiodic arrivals of transactions. Similar to the bursty and periodic arrival cases, GAS maintains 100% of DMR and negative L_{max} while FCFS and EDF have positive L_{max} and smaller DMRs than GAS. FCFS again has lower DMR and larger L_{max} than EDF due to its FCFS-based GTS scheduling method.



(a) DMR with 99% confidence intervals

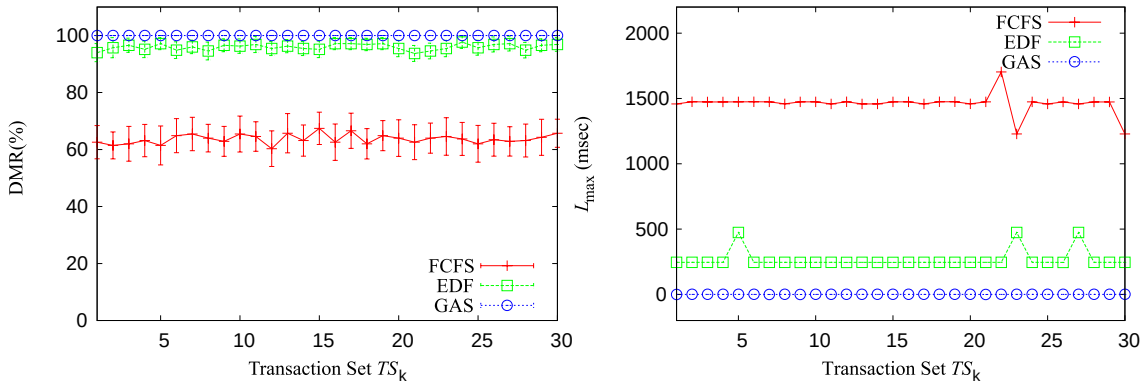
(b) L_{max}



(c) Utilization of GTS

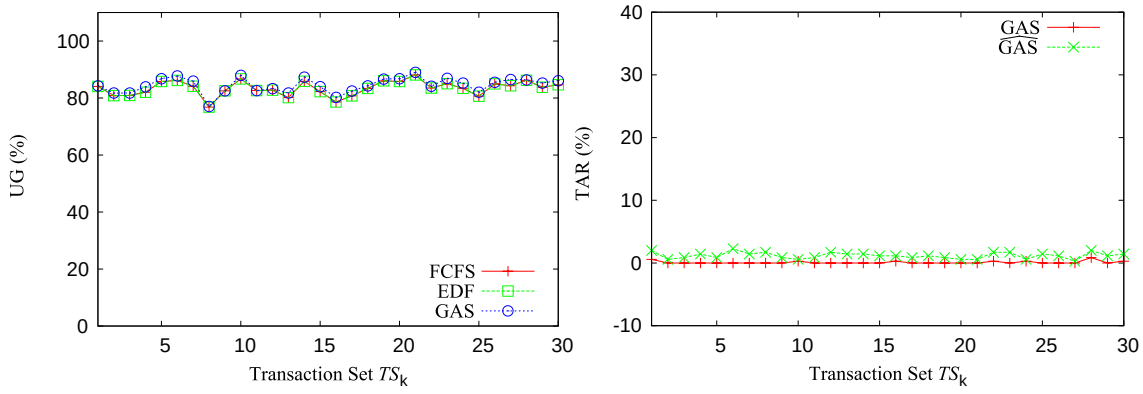
(d) TAR

Figure 3.16: Performance under periodic transmissions.



(a) DMR with 99% confidence intervals

(b) L_{max}



(c) Utilization of GTS

(d) TAR

Figure 3.17: Performance under aperiodic transmissions.

In Figure 3.17(c), the UG of GAS is comparable to the other algorithms and is slightly smaller than that of FCFS and Unfiform-EDF in some cases due to its rejection of infeasible transactions. Figure 3.16(d) demonstrates the benefits of Algorithm 3. Note that the UG of GAS in Figure 3.16(d) is lower than the UG of GAS in Figure 3.15(c). This is because GAS under aperiodic transactions must wait for the arrivals of the following transactions even if the current transaction is already completed, potentially lowering the GTS utilization.

3.9 Summary

In this chapter, we have presented a novel GTS allocation and scheduling algorithm, called GAS, to meet the delay constraints of delay-sensitive transactions in the IEEE 802.15.4 beacon-enabled star network. The GAS is proved to be optimal in *Theorem* 3.3 and work-conserving so that it achieves the maximal GTS utilization and can always find a feasible schedule if there exists a feasible schedule. Different from the EDF scheduling method [15], which results in bursty transmissions of payloads for transactions with delay constraints, it smooths out the traffic of a transaction by distributing the GTSs of a transaction over as many beacon intervals as possible while satisfying the delay constraint of the transaction.

By doing so, it reduces the average service starting time of transactions and enables concurrent services to more transactions. This can significantly benefit many delay-sensitive applications, where the starting time of the first message and the stability of traffic have great impact on the performance of these applications. Simulation results also demonstrate that the GAS significantly outperforms both the FCFS-based scheduling algorithm defined in the IEEE 802.15.4 and the EDF scheduling algorithm.

Our extensive simulation results demonstrate that the GAS has excellent performance under bursty, periodic and aperiodic transmissions of transactions. In particular, these results

indicate that the delay constraint meet ratio (DMR) of our algorithm is up to 100% higher than the FCFS-based scheduling algorithm. Our algorithm differs from the existing ones in that it is an on-line scheduling algorithm and allows transmissions of bursty and periodic messages with delay constraints even when the network is overloaded.

CHAPTER 4

MULTIRATE SERVICE DIFFERENTIATION

4.1 Introduction

Recent growing interest in the use of LR-WPAN has been driven by emerging applications such as home automation, health-care monitoring and environmental surveillance. To meet the needs for these applications, IEEE has announced a new standard called the IEEE 802.15.4 [1] for LR-WPAN. Similar to the Distributed Coordination Function (DCF) defined in the IEEE 802.11 [2], it defines a contention-based channel access mechanism called unslotted CSMA/CA channel access protocol. This protocol enables contending wireless sensors to access the shared channel without coordination by the centralized coordinator and, hence, is more flexible and robust for dynamic network formation than slotted CSMA/CA channel access mechanism based on centralized coordination, while it does not provide service differentiation at the MAC layer [30]. This lack of providing service differentiation in the IEEE 802.15.4 unslotted CSMA/CA protocol has hindered the development of service differentiation model for rate-sensitive applications.

To provide such differentiated service in wireless networks, the IEEE 802.11e [11] employs the Enhanced Distributed Coordination Function (EDCF), which consists of four Access Categories (ACs), Arbitrary Interframe Space (AIFS), $CW_{\min}$ and $CW_{\max}$. If one class has a smaller AIFS, $CW_{\min}$ or $CW_{\max}$, the class's traffic has a better chance to access the wireless

medium earlier. This standard is extended from the IEEE 802.11 DCF to support service differentiation while it does not have capabilities to provide rate-sensitive applications. This limitation is not desirable for LR-WPAN since rate-sensitive applications often demand QoS guarantees.

In the nonbeacon-enabled peer-to-peer network some operating scenarios for rate-sensitive applications arise when one considers wireless video surveillance and target detection applications for wireless sensor networks. To support such rate-sensitive applications in wireless sensor networks, we present the Multirate Service Differentiation (MSD) model operating in the nonbeacon-enabled peer-to-peer network. Unlike existing priority-based service differentiation models, the MSD defines the independent Virtual Medium Access Controls (VMACs), each of which consists of a transmission queue and the Adaptive Backoff Window Control (ABWC). The ABWC algorithm adjusts the backoff window to reflect the local network state in the local collision domain. Since the VMACs serve multiple rate-sensitive flows, it is possible that more than one data frame collide with each other when their backoff times expire simultaneously. To solve such a virtual collision in the virtual collision domain, we design the Virtual Collision Avoidance Control (VCAC). The VCAC algorithm prevents virtual collisions and preempts packets with the minimal cost in the virtual collision domain. By analyzing these algorithms, we prove that the ABWC component enables the achieved data rate to converge to the rate requirement and the VCAC component produces a virtual-collision-free schedule to avoid degradation of the achieved data rate. Through the simulation, we validate our analysis and show the MSD outperforms existing algorithms.

4.1.1 Existing Approaches

There has been plenty of research work focusing on QoS and service differentiation in wireless networks.

Some research work [30] [31] [32] [33] shows that the adjustment of backoff parameters

enables MAC protocols to support service differentiation. The research work [30] proposes an end-to-end service differentiation algorithm based on modified EDCF. Unlike the EDCF, it differentiates traffic class service depending on the forwarding rate of flow i and varies the backoff window with different weights. In [32] [33], the researchers present a backoff-based priority schemes to provide service differentiation by varying three backoff parameters such as initial window size $W_{i,0}$, window-increasing factor σ_i and retry limit $L_{i,\text{retry}}$ in terms of the priority i class. In [31], two service classes for delay-sensitive and best-effort traffics are defined, each of which is assigned different CW_{min} and CW_{max}, where CW_{min} and CW_{max} for delay-sensitive traffics is lower than those of best-effort traffics. It is shown that this algorithm provides good service differentiation in terms of throughput and delay over high-priority and best-effort traffics. The focus of all these approaches is on simply assigning different backoff parameters to traffic classes or ACs while they have a significant limitation in achieving QoS guarantees.

Some existing work [34] [35] [36] places a focus on controlling over the contention window depending on the changes in the network status. In [34], the Sensing Backoff Algorithm (SBA) has been addressed to maximize channel throughput with fair access to a shred channel. On a packet collision, it multiplies its backoff interval by α while on a successful transmission, both sending and receiving wireless sensors multiply their backoff interval by θ, and the others overhearing(sensing) a successful transmission decrease their backoff intervals by β. Cali *et al.* address a dynamic IEEE 802.11 [36] [35] based on p-persistent CSMA/CA protocol. To compute the accurate network status, however, their approach must estimate the exact number of active wireless sensors while they do not consider QoS for real-time traffics since they focused on fair access to the shared channel.

Different from MAC-level service differentiation, transport-level QoS called SWAN [37] has been proposed. Based on AIMD window control algorithm, SWAN assures the bandwidth for real-time traffics as much as possible while regulating the bandwidth for best-effort traffics.

Since it lacks the QoS support from the underlying MAC protocol, it must resort to binary exponential backoff mechanism when transmitting real-time traffics, which results in frequent violations of QoS requirements. This is because SWAN controls over the transmission rate only on top of the MAC layer.

Recent research works in IEEE 802.15.4 WSNs have presented priority-based service differentiation models similar to EDCF. Because of no definition of variable Interframe Spaces (IFSs) such as AIFS, these works are based on a simple scheme that assigns different Backoff Exponent (BE) to service classes. The authors in [38] [39] proposed a multi-level service differentiation model based on a single prioritized queue. Depending on priority p, each service class is assigned the initial BE, i.e., $BE[p]$ and the backoff period is given by one of the random values from $[2^{(BE-1)}, 2^{BE} - 1]$ whenever a collision occurs.

Unlike [38] [39], Koubaa *et al.* in [40] presented a service differentiation model with prioritized multiqueues. Their algorithm gives each service class two parameters, i.e., the minimal BE and the maximum BE. On a collision, the backoff period is given by a random value from $[2^{SC_i(BE)}, 2^{SC_i(maxBE)}]$, where SC_i is service class i and BE increases by 1. However, their algorithm does not allow lower-priority packets to be preempted during the backoff in spite of arrival of higher-priority packets. Kipnis *et al.* in [41] designed a prioritized multiqueue-based service differentiation model operating on top of the IEEE 802.15.4 MAC layer. In their model, higher-priority packets are always selected to be transmitted by preempting lower-priority packets during the backoff. As a result, their approach incurs no virtual collision.

With no consideration about the rate requirements of data flows, all these algorithms aim at simply maximizing the throughput and providing such flows with a level of service depending on priorities. In other words, a service class with a higher priority has more opportunities to occupy the wireless channel than one with a lower priority. This introduces a problem that higher-priority data flows occupy the wireless channel more than what they need for meeting

their minimal rate requirements.

4.1.2 Chapter Organization

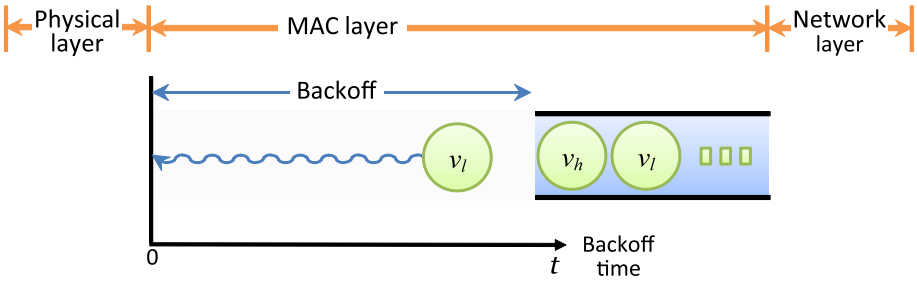
The remainder of this chapter is organized as follows. In Section 4.2 we address the problems of existing service differentiation. In Section 4.3 we present the design of the MSD model including the design of VMAC. Section 4.4 presents the design of the ABWC and VCAC algorithms. In Section 4.5 we analyze these two algorithms mathematically. In Section 4.6 we explain how to implement the MSD on the IEEE 802.15.4-enabled wireless sensor. The performance evaluation, including the experimental results, is given in Sections 4.7 and 4.8. Finally, we give a summary of the chapter in Section 4.9.

4.2 Problem Statement

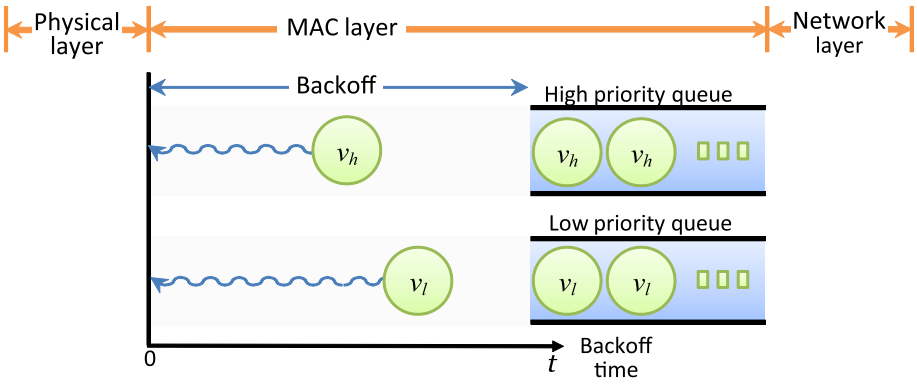
Existing priority-based service differentiation models based on the IEEE 801.15.4 fall into Single Priority Queue (SPQ) [38] [39] [40] and Multiple Priority Queues (MPQ) [41] [11].

The SPQ model has a single priority queue, where a data frame is assigned either a high priority or a low priority. With this model, a data frame with a high priority is assigned a shorter backoff time than one with a low priority since it has a lower BE than the low priority frame.

If the low priority frame is being in backoff, however, the high priority frame must wait until the low priority frame is transmitted. This raises a significant problem that the data frame with a high priority fails to meet its rate requirement. Since the SPQ model sends data frames in the First-In-First-Out (FIFO) manner regardless of priority, the SPQ model does not provide any guarantee for rate-sensitive applications so that it is not suitable for rate-sensitive applications.



(a) Single priority queue



(b) Multiple priority queues

Figure 4.1: Two service differentiation models: Single Priority Queue (SPQ) and Multiple Priority Queues (MPQ). v_l is an instance of the data frame with a low priority. v_h is an instance of the data frame with a high priority.

On the other hand, the MPQ model has multiple priority queues, each of which is assigned a priority and stores data frames with the same priority. With this model, data frames in a high priority queue are assigned a low BE while data frames in a low priority queue are assigned a high BE. Namely, the high priority frames can be sent before the low priority frames since they backoff independently. Such an independent backoff introduces a significant problem called a virtual collision if the backoff times of two data frames have expired simultaneously. The Virtual Collision Handler (VCH) [11] is defined to solve this problem. It chooses a data frame with the highest priority among all instances being in backoff and forces the others to backoff again. In [41] no virtual collision occurs since high priority frames preempts low priority frames when they are being in backoff.

Such frame preemption in the MAC layer frequently happens during the backoff since high priority frames are always transmitted prior to low priority frames even though these low priority frames are already being in backoff. It introduces a problem that if a low priority frame, v_l is in its first backoff and a high priority frame, v_h restarts its second backoff due to a local collision, then v_l will be preempted by v_h even though v_l's backoff time nearly has reached 0 and v_l's required data rate has not been missed. As a result, v_l is forced to backoff again and its achieved data rate will be degraded when it is transmitted without interruption by high priority data frames. In other words low priority frames suffer from lack of transmission opportunity.

In addition, it is not necessarily desirable to send a higher priority frame than a lower priority frame in real-time communication scenarios. For example, consider a high priority frame that suffers from a significant latency in its previous hops and hence will not be useful even if it arrives at the destination. In such a case, it is better to transmit a lower priority packet since this packet will yield more benefits than the high priority frame that has already missed its rate requirement.

Suppose that in Figure 4.2 there are three instances, v_1 with p_1 and r_1^*, v_2 with p_2 and r_2^*,

and v_3 with p_3 and r_3^*, where p_1(highest)>p_2>p_3(lowest) and r_1^*, r_2^*, and r_3^* are the rate requirements. At t_1, v_1 starts backoff and is transmitted when its backoff time expires. In this case, there is no virtual collision since only v_1 is being in backoff. If v_1 collides with any other frame in the local network, it will backoff again so that it is assigned a longer backoff time, t_2, as shown in the Figure. At t_2, v_1, v_2 and v_3 start backoff simultaneously. Note that v_2 and v_3 are being in their first backoff. It is obvious that these three instances collide with one another in the virtual collision domain when their backoff times have expired at $t = 0$ simultaneously. When such a virtual collision occurs, the VCH chooses v_1 (since it has the highest priority, p_1) even though v_1 has already missed the rate requirement.

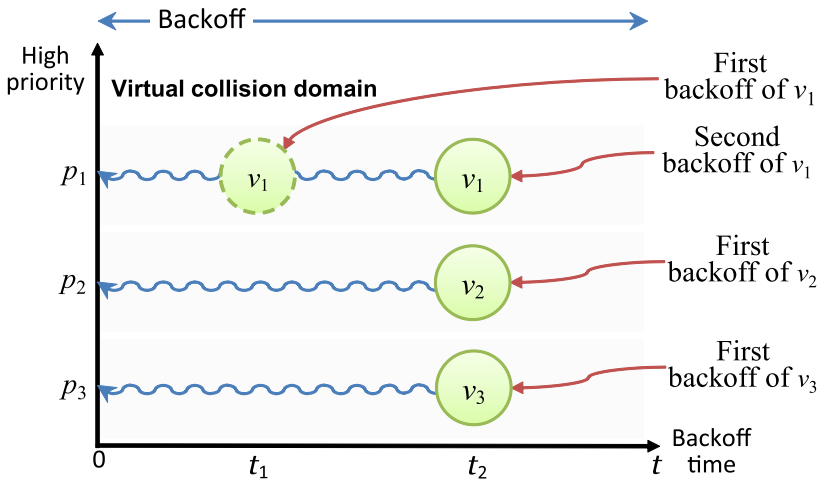


Figure 4.2: Problem of virtual collision when three backoff times have expired simultaneously. p_1(highest)>p_2>p_3(lowest). t_1(earliest)<t_2(latest). At $t = 0$, the backoff times of v_1, v_2 and v_3 expire and the virtual collision occurs.

On the other hand, v_2 and v_3 are forced to backoff again even though they do not miss their rate requirements yet. As a result, all of the three instances will miss their rate requirements. It is not necessarily desirable to send the higher priority frame than the lower priority frame in real-time communication scenarios if the high priority frame has already missed its rate requirement. In such a case, it is better to transmit either v_2 or v_3 rather than v_1 since one the low priority frames will yield more benefits than v_1 that has already missed its rate

requirement.

In the following section we present the MSD model that not only provides guaranteed service for multiple data flows with rate requirements, but also solve the virtual collision and handling problems.

4.3 MSD Model

4.3.1 Network Model and Assumptions

We consider the IEEE 802.15.4 nonbeacon-enabled peer-to-peer network based on unslotted CSMA/CA channel access mechanism. All wireless sensors are not synchronized with one another and thus transmissions are not aligned with slot boundaries. Furthermore, a general wireless network is taken into consideration, i.e., all wireless sensors are not necessarily in their transmission range. For simplicity, we assume that a wireless sensor i can receive a packet from a wireless sensor j if and only if the wireless sensor j can receive a packet from the wireless sensor i. We also assume that all wireless sensors are stationary and share a single wireless channel. Furthermore, they have a single transceiver so that they cannot transmit and receive simultaneously.

4.3.2 Virtual Medium Access Control

For different types of flows, we consider the Virtual Medium Access Control (VMAC), which is an independent MAC entity consisting of a transmission queue and an Adaptive Backoff Window Control (ABWC) component. Figure 4.3 shows the MSD model that contains VMAC entities and VCACs. Let $\mathcal{M}$ be a set of active VMAC entities of a wireless sensor. It is represented as $\mathcal{M} = \{z_i : 1 \leq i \leq m\}$.

The Network State Estimator (NSE) measures the network congestion level and quantifies how many collisions occur in the LCD. Each VMAC entity receives this information such that it is able to dynamically adjust its backoff window size by means of the ABWC. The Virtual Collision Avoidance Control (VCAC) algorithm is responsible for preventing virtual collisions incurred by simultaneous transmission attempts of more than two VMAC entities in the VCD. It produces a virtual collision-free schedule consisting of the instances of VMAC entities. The details of the VCAC algorithm are explained in section 4.4.3.

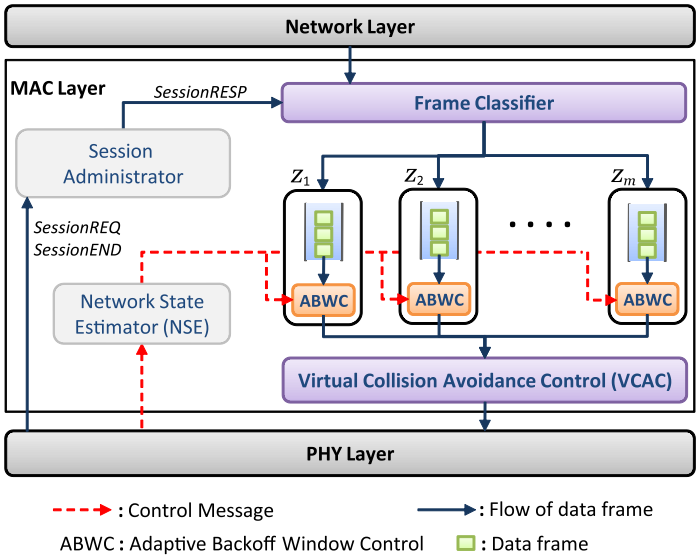


Figure 4.3: The MSD model with VMAC and VCAC.

4.3.3 Admission Control and Session Establishment

The session administrator creates a new session with a VMAC entity if there are enough resources to serve the rate requirement. Note that z_1 is reserved for best-effort flows. Since best-effort flows are tolerant to changes in service quality levels and do not have any strict rate requirements for packet delays, they are queued into z_1 while rate-sensitive flows are sent to one of the other VMAC entities.

A per-flow session is established by Forward Resource Reservation Protocol (FRRP). To establish a new data flow session, for example, a source wireless sensor sends a SessionREQ

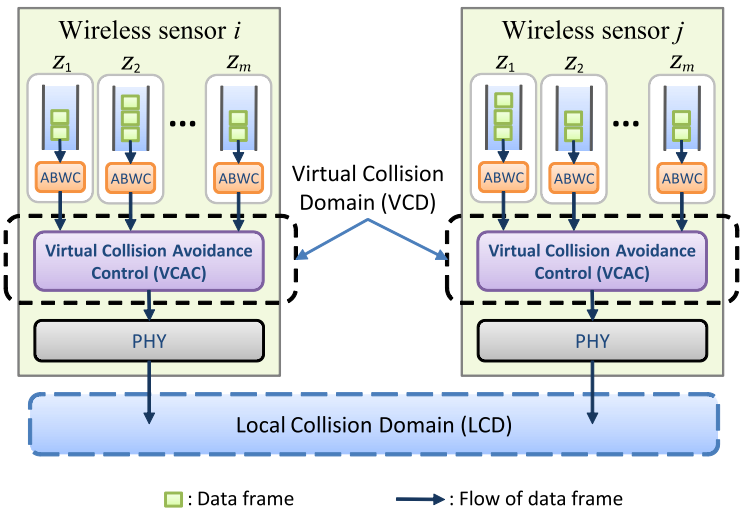


Figure 4.4: Virtual collision domain vs. local collision domain.

message to a destination. If wireless sensors on the source/destination path have enough resources to serve, these resources are reserved until they are not needed any more. The destination responds with a SessionRESP message over the backward path. If there are not sufficient resources, they send a SessionRESP message with a $NotAvailableResource$ bit so that the reserved resources are released.

Once the dat flow session is established on the source/destination path, the source is allowed to send packets. On the other hand, the data flow session ends either by sending a SessionEND message explicitly or by the expiration of a session timer when there is no data flow. We assume that the timer value is large enough to detect nonexistence of data flow over the data flow session.

4.4 Algorithm Design

In this section, we discuss how to adjust the backoff window size to meet the required data rate for rate-sensitive data flows depending on the network behavior in the LCD. Since a wireless sensor is allowed to establish multiple sessions, it is inevitable that they contend

with one another to get an opportunity for exclusive use of the channel in the VCD. In the following section, we explain how to schedule instances from those sessions to avoid virtual collisions caused by simultaneous transmission attempts of more than two instances.

4.4.1 Adaptive Backoff Window Control

The main factor that impacts significantly on the data rate is the backoff delay, which varies depending on backoff windows. To fulfill the rate requirement and realize the distributed nature of backoff window control, we present the ABWC algorithm. It dynamically adjusts the backoff window at every transmission to achieve the rate requirement. For $z_i \in \mathcal{M}$, it is formulated as:

$$\begin{cases} w[z_1] = \mathsf{random}\left(2^{\mathsf{min}\{BE[z_1], macMaxBE\}} - 1\right) + f_{\text{max-bw}}, \\ w^{k+1}[z_i] = w^k[z_i] + \theta[z_i]\Psi_{\text{nsm}}(r^k[z_i] - r^*[z_i]),\ (i > 1 \text{ and} \\ \text{ if } (r^k[z_i] - r^*[z_i] < 0) \text{ and } (\Psi_{\text{nsm}} > 1), \text{ then } \Psi_{\text{nsm}} = -\Psi_{\text{nsm}}), \end{cases} \tag{4.1}$$

where $\mathsf{random}(\cdot)$ returns an integer with a uniform distribution, $w^k[z_i]$ is the kth update of $w[z_i]$'s backoff window, $r^*[z_i]$ is the rate requirement of z_i, $\theta[z_i]$ is the constant scaling factor, BE stands for Backoff Exponent, which takes one of the integers from [*macMinBE,macMaxB E*], and $r[z_i]$ is the achieved data rate. $r[z_i]$ is represented as:

$$r[z_i] = \frac{L[z_i]}{w[z_i]\delta + \Delta t_{\text{ta}}},\ (z_i \in \mathcal{M} \backslash \{z_1\}), \tag{4.2}$$

where $L[z_i]$ is the length of a packet transmitted by z_i, δ is a unit backoff time and Δt_{ta} is a duration, which includes the Clear Channel Assessment (CCA) [1], transmission delay, propagation delay and Ack.

Ψ_{nsm} is the network state estimator consisting of f_{cli} and f_{nci} and returns the sum of two estimated values of network state. It is given by:

$$\Psi_{\text{nsm}} = f_{\text{cli}} + f_{\text{nci}} + 1, \tag{4.3}$$

where f_{cli} and f_{nci} are congestion level indicator and network collision level indicator respectively.

f_{cli} is the congestion level indicator and returns a level of current network congestion. It is given by:

$$f_{\text{cli}} = \gamma_{\text{cli}} \frac{n_{\text{busy}}}{n_{\text{smpl}}}, \ (\gamma_{\text{cli}} > 0), \tag{4.4}$$

where γ_{cli} is a small positive constant, n_{smpl} is the number of channel states periodically sampled and n_{busy} is the number of busy states out of n_{smpl}. If $f_{\text{cli}} > 0$, it indicates that local congestion is not negligible and the backoff window needs to be increased. Otherwise, f_{cli} indicates that local congestion is negligible.

f_{nci} is the network collision indicator which means how many collisions occur in the LCD. It is given by:

$$f_{\text{nci}} = \gamma_{\text{nci}}(NB[z_i] - 1), \ (\gamma_{\text{nci}} > 0), \tag{4.5}$$

where γ_{nci} is a positive constant and $NB[z_i]$ is the number of backoffs performed by z_i. On a collision, NB increases by 1 until it reaches $macMaxCSMABackoffs$ [1]. If $f_{\text{nci}} = 0$ (which means that current local collision is negligible), then it leads to no increase in the backoff window. On the other hand, if $f_{\text{nci}} > 0$, then it leads to increase in the backoff window since current local collision is not negligible.

$f_{\text{max-bw}}$ returns the largest backoff window of active VMAC entities. It is given by:

$$f_{\text{max-bw}} = w[z_\lambda] + \left\lceil \frac{d[z_\lambda]}{\delta} \right\rceil, \tag{4.6}$$

where $d[z_\lambda]$ is the sum of transmission delay and CCA duration.

$d[z_i]$ is the elapsed backoff time, which is represented as:

$$d[z_i] = \Delta t_{\mathrm{td}}[z_i] + \Delta t_{\mathrm{cca}}, \tag{4.7}$$

where $\Delta t_{\mathrm{td}}[z_i]$ is the transmission delay achieved by z_i and Δt_{cca} is the CCA duration. λ is an index of the VMAC entity that has the largest w and is defined by:

$$\lambda \triangleq \mathsf{argmax}_{2 \le i \le m} \{w[z_i]\}. \tag{4.8}$$

Since best-effort flows have no rate-sensitive properties, $w[z_1]$ always has the largest backoff window out of active $w[z_i]$ to prevent z_1's transmission from incurring local collisions, while the other VMAC entities adjust their $w[z_i]$ depending on variations of both $r[z_i]$ and Ψ_{nsm}. Note that despite being regarded as best-effort flows, link layer protocols (e.g., ARP) or routing protocols (e.g., AODV) are treated as flows with the highest priority and cost since they require timely responses.

$\theta[z_i]$ is the scaling factor, which is is given by:

$$\theta[z_i] = \gamma_{\mathrm{bw}} \frac{r^*[z_i]}{\sum_{z_k \in \mathcal{M} \setminus \{z_1\}} r^*[z_k]}, \ (\gamma_{\mathrm{bw}} > 0, z_i \in \mathcal{M} \setminus \{z_1\}), \tag{4.9}$$

where γ_{bw} is a scaling factor.

4.4.2 Cost Function Model

As stated in Section 4.2, the VCH chooses an instance with the highest priority among all instances being in backoff and forces the others to backoff again. It is not necessarily desirable to choose the highest priority instance out of the instances when the virtual collision occurs. For example, consider a high priority frame that suffers from a significant latency in its

previous hops and hence will not be useful even if it arrives at the destination. In such a case, it is better to transmit a lower priority packet since this packet will yield more benefits than the high priority frame that has already missed its rate requirement.

To solve this problem, we define the cost function $C(t)$, which returns a cost value with regard to time t. Basically, when one instance needs to be chosen out of the instances, the cost function for each instance returns $C(t)$ with regard to t. It is given by:

$$C(t) = \begin{cases} C^{\min} & \text{if } r^* = 0, \\ C^{\max} & \text{if } t \in [0, t^*], \\ C^{\max} - \alpha(t - t_c) & \text{if } t > t^*, \end{cases} \tag{4.10}$$

where $C^{\min}$ and $C^{\max}$ is the minimum and maximum of cost values, r^* is the rate requirement and $\alpha > 0$. $r^* = 0$ indicates best-effort flows since they are regarded as less significant than rate-sensitive flows and hence their $C(t)$ always returns $C^{\min}$. $C(t)$ maintains a constant until $C(t)$ reaches $C(t^*)$ and linearly decreases after t^*.

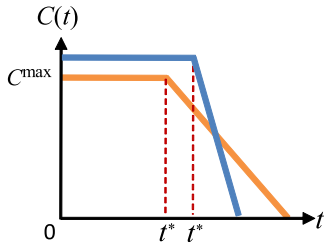


Figure 4.5: Cost function $C(t)$.

Intuitively, this implies that the instance of a data frame is considered to be important until $C(t)$ reaches $C(t^*)$ while it is considered to be less important after $C(t^*)$. Suppose that v_1's cost function is $C_1(t)$ and v_2's cost function is $C_2(t)$. If a virtual collision occurs and $C_1(t) > C_2(t)$ at t, then v_1 is more important than v_2 so that v_2 will be chosen. As shown in the Figure 4.4.2, $C(t)$ is decreasing after t^*. This is because strict discard of data frames after t^* may increase the rate of dropping frames. In Section 4.4.3, we discuss how the cost function is used by the VCAC.

4.4.3 Virtual Collision Avoidance Control

The existing virtual collision resolution algorithms such as [11] simply choose one of the colliding instances with the highest priority in the VCD and allow it to be transmitted while the other colliding instances perform the backoff procedure with increased backoff window sizes for next transmission round. Since this simple algorithm chooses only one with the highest priority on the virtual collision, the other colliding instances must have longer backoff times so that not only is their transmission delayed, but also their data rates are degraded (See Equation 4.2).

In this section two virtual collisions are considered: (1) more than two instances' backoff timers are expired simultaneously, and (2) for two instances v_p and v_q, during either v_p's transmission or CCA, v_q's backoff timer expires and vice versa. Clearly, the first case accounts for a virtual collision. The second case is also considered to be a virtual collision since v_p and v_q collide with each other before their transmission either if they are trying to sense the channel simultaneously by means of CCA or if v_p is being transmitted and v_q is trying to sense the channel simultaneously. In these two cases, only one VMAC entity must be allowed to access the channel and the others backoff again. To avoid virtual collisions and prevent degradation of the data rate, we present a new virtual collision resolution algorithm, called VCAC, which does not degrade the required data rates of those instances.

Figure 4.6 shows a sequence of function calls in the VCAC. Firstly, VirtualCollisionAvoidance Control() invokes ProduceEBTFSch−edule(), which makes a new $\mathcal{V}$ in the EBTF order. ProduceEBTFSchedule() adds a new instance to $\mathcal{V}$ and invokes PredictVirtualCollision(), which checks if any instance in $\mathcal{V}$ causes a virtual collision. If there is no virtual collision, then PredictVirtualCollision() invokes ReallocateBackoffTime(), which adjusts the backoff time of each instance in $\mathcal{V}$ if reallocating the backoff time is needed. Otherwise, it invokes ReplaceInsta nceWithMinimalCost(), which chooses one instance with the minimal cost and replaces it with the new instance. Since the placement of instances in $\mathcal{V}$ has changed, it invokes

PredictVirtualCollision() again to check the virtual collision. If the virtual collision still occurs, then PredictVirtualCollision() invokes Backoff(), which removes the instance (being replaced with the new instance) if the instance's NB reaches $macMaxCSMABackoffs$. Otherwise, the backoff window of the instance is updated in accordance with the ABWC in Equation (4.1).

In the following we explain more details of how the VCAC algorithm works. Basically, the VCAC algorithm inserts a new instance and reallocates instances' backoff times when necessary. It consists of Algorithm 7 through Algorithm 12. Algorithm 7 calls ProduceEBTFSchedule() in Algorithm 8 and ReplaceInstanceWithMinimalCost() in Algorithm 9. As the name implies, ProduceEBTFSchedule() produces a virtual collision-free schedule, whose the instances are ordered in the EBTF.

Firstly, it looks into the feasibility condition, for which PredictVirtualCollision() in Algorithm 10 is responsible. PredictVirtualCollision() finds out a place where v_{new} will be added and checks if $t_{\text{bo}}[v_{\text{new}}]$ is greater than t_{sum}, which is the sum of $d[v_j]$. If $t_{\text{bo}}[v_{\text{new}}] > t_{\text{sum}}$ (which means there is enough space to add v_{new}), then the schedule turns out to be feasible.

Secondly, ProduceEBTFSchedule() updates instances' backoff times, for which ReallocateBackoffTime() in Algorithm 11 is responsible, if the schedule is feasible. ReallocateBackoffTime() reallocates a new backoff time shorter than current backoff time if PredictVirtualCollision() in Algorithm 10 detects the likelihood of a virtual collision between two successive instances. The reallocation of new backoff times may lead to a temporary increase in the data rate. However, the ABWC enables the increased data rate to decrease (See Equation (4.1)).

If ProduceEBTFSchedule() fails to produce a feasible schedule with v_{new} (which implies that there is not enough space to add it), then the VCAC chooses one of the instances. The

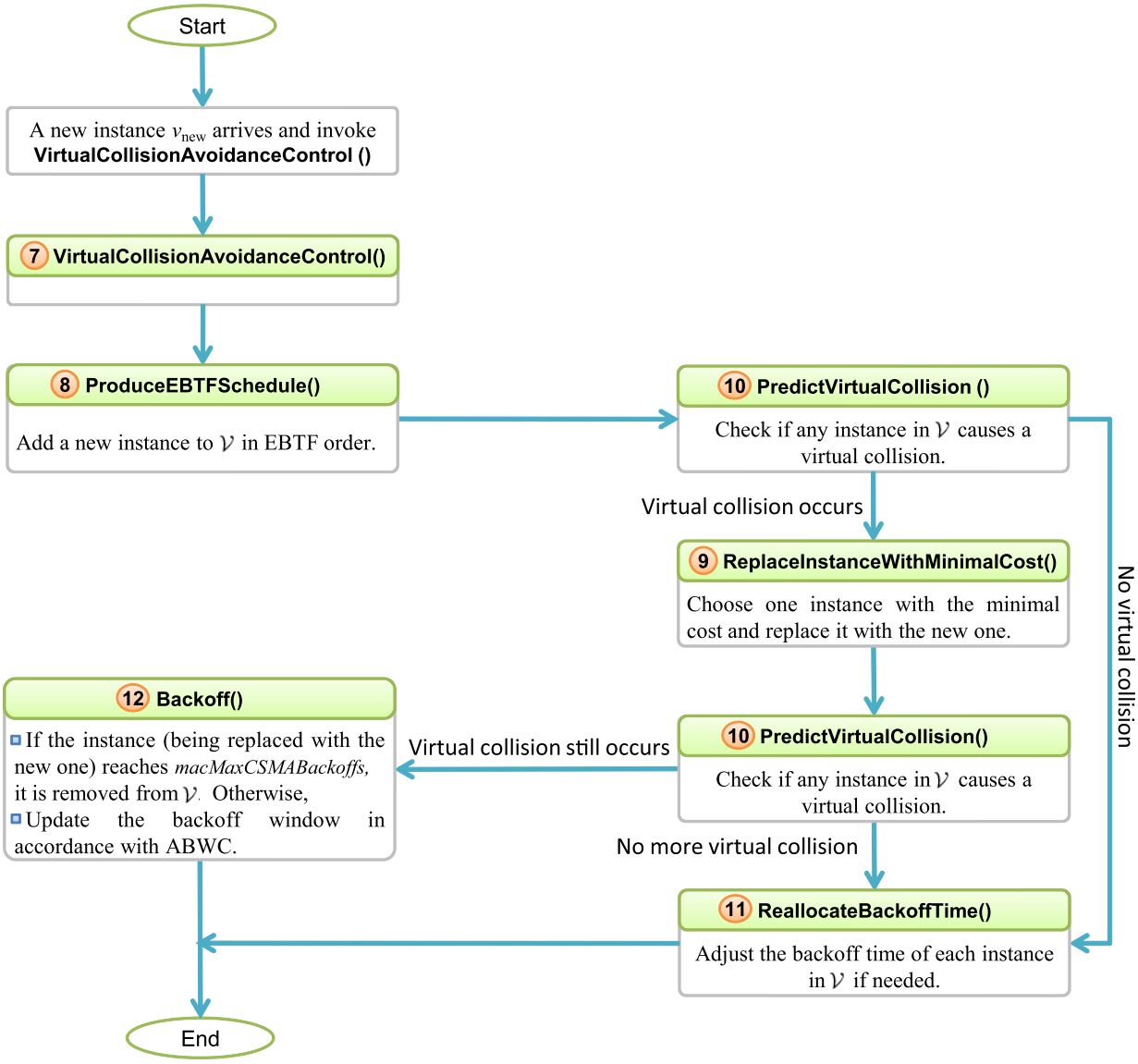


Figure 4.6: Control flow of the VCAC.

Algorithm 7: VirtualCollisionAvoidanceControl()

```
Input  : v_new, V
Output: none
if ProduceEBTFSchedule(v_new, V) = fail then
    if ReplaceInstanceWithMinimalCost(v_new, V) = fail then
        Backoff(v_new)
        Reschedule v_new.
    end
end
```

Algorithm 8: ProduceEBTFSchedule()

Input : $v_{\text{new}}, \mathcal{V}$
Output: *success* or *fail*
Data: j
if $r^*[v_{\text{new}}] = 0$ **then**
    Link v_{new} *to the last instance of* $\mathcal{V}$.
    return *success*
end
if PredictVirtualCollision$(v_{\text{new}}, |\mathcal{V}|, \mathcal{V}) = fail$ **then return** $fail$
Add v_{new} *to* $\mathcal{V}$ *in order of nondecreasing* $t_{bo}[v_j]$.
ReallocateBackoffTime$(|\mathcal{V}|, \mathcal{V})$
return *success*

Algorithm 9: ReplaceInstanceWithMinimalCost()

Input : $v_{\text{new}}, \mathcal{V}$
Output: *success* or *fail*
Data: j, k, v_{tmp}
Find the index k *of the instance with the minimal cost value in* $\mathcal{V}$.
if $k = |\mathcal{V}|$ **then** $v_k \leftarrow v_{\text{new}}$
else if $k > 0$ **then**
    Assign v_k *to* v_{tmp} *and* $d[v_{new}]$ *to* $d[v_k]$.
    if PredictVirtualCollision$(v_k, k+1, \mathcal{V}) = fail$ **then**
        Assign v_{tmp} *to* v_k *and return fail.*
    end
    ReallocateBackoffTime$(k+1, \mathcal{V})$
end
return *success*

Algorithm 10: PredictVirtualCollision()

Input : $v_{\text{new}}, y, \mathcal{V}$
Output: *success* or *fail*
Data: $j, t_{\text{sum}}, Backoff$
Sum up $t_{\text{bo}}[v_j]$ *until* $t_{\text{bo}}[v_{\text{new}}] < t_{\text{bo}}[v_j]$ *for* $j \in [1, y]$, *and assign it to* t_{sum}.
if $t_{\text{bo}}[v_{\text{new}}] < t_{\text{bo}}[v_y]$ **then** $Backoff \leftarrow t_{\text{bo}}[v_{j+1}] - d[v_{\text{new}}]$
else $Backoff \leftarrow t_{\text{bo}}[v_{\text{new}}]$
if $Backoff > t_{\text{sum}}$ **then**
    Assign $Backoff$ *to* $t_{bo}[v_{new}]$ *and return success.*
end
return *fail*

Algorithm 11: ReallocateBackoffTime()

Input : $y, \mathcal{V}$
Output: None
Data: j
for $j \leftarrow y$ to 2 **do**
    if $w[v_j] > w[v_{j-1}]$ and $t_{\text{bo}}[v_j] - t_{\text{bo}}[v_{j-1}] > d[v_{j-1}]$ **then**
        continue
    end
    else if $w[v_j] > w[v_{j-1}]$ and $t_{\text{bo}}[v_{j-1}] - t_{\text{bo}}[v_j] > d[v_j]$ **then**
        continue
    end
    else $t_{\text{bo}}[v_{j-1}] \leftarrow t_{\text{bo}}[v_j] - d[v_{j-1}]$
end

Algorithm 12: Backoff()

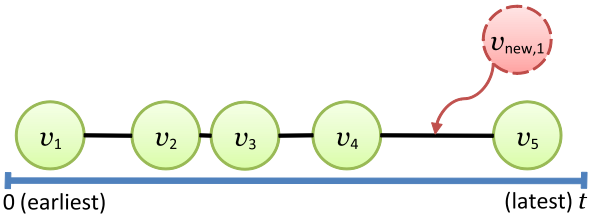
Input : v
Output: none
if $NB[v]+1 > macMaxCSMABackoffs$ **then**
    Remove v from $\mathcal{V}$.
    return
end
else if $r^*[v]=0$ **then** $w[v] \leftarrow \text{random}\left(2^{\min\{BE[z_1], macMaxBE\}}-1\right) + f_{\text{max-bw}}$
else $w[v] \leftarrow w[v] + \theta[v]\Psi_{\text{nsm}}\left(r[v]-r^*[v]\right)$
$NB[v] \leftarrow NB[v]+1$
if $BE[v] < macMaxBE$ **then** $BE[v] \leftarrow BE[v]+1$
Update the cost value $U[v]$ *in accordance with equation* (4.10).

selected one has the smallest $C(t)$ less than v_{new}'s $C(t)$. Then, the VCAC replaces it with v_{new}.

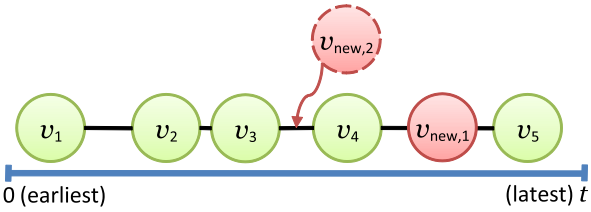
Note that the instance with $r^*[v]=0$ has the smallest $C(t)$ if the schedule includes it. If the schedule has no instance less than v_{new}'s $C(t)$, the new instance backoffs again. Backoff() in Algorithm 12 does such a backoff. Note that the VCAC does not guarantee successful transmission in the LCD, but reduces potential degradation of the data rate by avoiding virtual collisions in the VCD. For simplicity we denote $C(t)$ as c_t, which is a cost value at t.

Figure 4.7 shows an example of a schedule produced by the VCAC algorithm. We assume that the initial schedule includes five instances and they satisfy both $c_t[v_{\text{new},1}] = c_t[v_{\text{new},2}] = c_t[v_{\text{new},3}] > c_t[v_5] > c_t[v_4] > c_t[v_3] > c_t[v_1] > c_t[v_2]$ and $t_{\text{bo}}[v_1] < t_{\text{bo}}[v_2] < t_{\text{bo}}[v_3] < t_{\text{bo}}[v_{\text{new},3}] < t_{\text{bo}}[v_{\text{new},2}] < t_{\text{bo}}[v_4] < t_{\text{bo}}[v_{\text{new},1}] < t_{\text{bo}}[v_5]$. Note that v_1 is the earliest while v_5 is the latest.

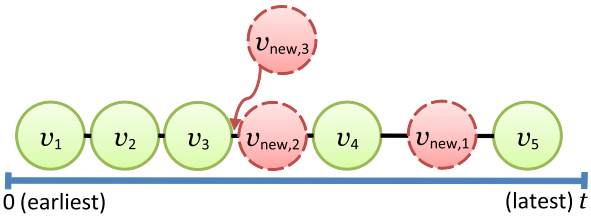
Figure 4.7(a) illustrates the schedule before adding a new instance, $v_{\text{new},1}$. When $v_{\text{new},1}$ arrives, the VCAC adds it to the schedule since the schedule not only satisfies $U_{\text{BT}} < 1$ in accordance with Algorithm 10, but also has enough time space without reallocating the instances' backoff times. Figure 4.7(b) shows that $v_{\text{new},1}$ has been added successfully.



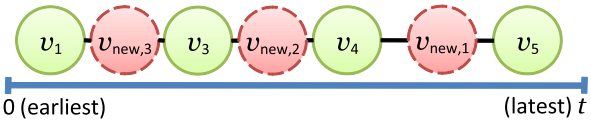
(a) Initial schedule and $v_{\text{new},1}$ arrives.



(b) $v_{\text{new},1}$ is added and $v_{\text{new},2}$ arrives.



(c) $v_{\text{new},2}$ is added and $v_{\text{new},3}$ arrives.



(d) Completed schedule.

Figure 4.7: Virtual-collision-free schedule produced by the VCAC.

In Figure 4.7(b), the VCAC attempts to add $v_{\text{new},2}$ to the schedule since $t_{\text{bo}}[v_3] < t_{\text{bo}}[v_{\text{new},2}] < t_{\text{bo}}[v_4]$. Since the schedule not only has enough time space and but also satisfies $U_{\text{BT}} < 1$, the VCAC adds it to the schedule. At this time, Algorithm 10 detects virtual collisions caused by $v_{\text{new},2}$ so that the VCAC reallocates the backoff times of those instances earlier than $v_{\text{new},2}$ to prevent the virtual collision. These reallocated backoff times of v_2, v_3 and $v_{\text{new},2}$ are less than their current backoff times since Algorithm 11 allocates shorter backoff times to avoid the virtual collision. This implies that the VCAC does not make those instances's backoff delays longer so that it reduces potential degradation of their data rates.

Figure 4.7(c) shows that the shorter backoff times of v_2, v_3 and $v_{\text{new},2}$ have been reallocated. Clearly, these shorter backoff times increase the data rates of v_2, v_3 and $v_{\text{new},2}$. However, the data rate will decrease as much as it increases in accordance with the ABWC (See Equation 4.1).

In Figure 4.7(c), the VCAC attempts to add $v_{\text{new},3}$ to the schedule since $t_{\text{bo}}[v_3] < t_{\text{bo}}[v_{\text{new},3}] < t_{\text{bo}}[v_{\text{new},2}]$. When it is inserted, the schedule turns out to be infeasible since $U_{\text{BT}} > 1$. Since v_2 has the smallest cost value, the VCAC replaces it with $v_{\text{new},3}$ in accordance with Algorithm 9 and makes it backoff again for next transmission round. On the other hand, if $v_{\text{new},3}$ does not meet the feasibility condition in spite of having replaced v_2, the VCAC has $v_{\text{new},3}$ backoff again and puts back v_2 where it was (See lines 12-18 in Algorithm 9).

Figure 4.7(d) shows the completed schedule right after the VCAC has finished adding $v_{\text{new},1}$, $v_{\text{new},2}$ and $v_{\text{new},3}$.

4.5 Algorithm Analysis

In this section, we analyze both ABWC and VCAC algorithms explained in section 4.4 and elaborate their mathematical and theoretical properties.

4.5.1 Analysis of ABWC Algorithm

Theorem 4.1 *(Convergence) For $z_i \in \mathcal{M} \backslash \{z_1\}$, $r[z_i]$ converges to $r^*[z_i]$.*

Proof. We show that with regard to $w[z_i]$, a *Lyapunov* function exists so that $w[z_i]$ is stable. Let $V(w[z_i]) = [\Psi_{\text{nsm}}(r[z_i] - r^*[z_i])]^2 = \left[\Psi_{\text{nsm}}\left(\frac{L[z_i]}{w[z_i]\delta + \Delta t_{\text{ta}}} - r^*[z_i]\right)\right]^2$ so that $V(w[z_i])$ is continuously differentiable and positive definite for $w[z_i] \neq w^*[z_i]$, where $V(w^*[z_i]) = 0$. We denote the time derivative of Equation (4.1) by $\dot{w}[z_i] = \theta[z_i]\Psi_{\text{nsm}}(r[z_i] - r^*[z_i])$. Then, the time derivative of $V(w[z_i])$ is calculated as:

$$\begin{aligned} \dot{V}(w[z_i]) &= \frac{\partial V}{\partial w[z_i]} \dot{w}[z_i] \\ &= -\frac{2\delta L[z_i]\Psi_{\text{nsm}}}{(w[z_i]\delta + \Delta t_{\text{ta}})^2} (r[z_i] - r^*[z_i])\, \dot{w}[z_i] \\ &= -\frac{2\delta\theta[z_i]r[z_i]}{w[z_i]\delta + \Delta t_{\text{ta}}} [\Psi_{\text{nsm}}(r[z_i] - r^*[z_i])]^2 \end{aligned} \tag{4.11}$$

It it obvious that $\dot{V}(w[z_i]) < 0$ for $w[z_i] \neq w^*[z_i]$, that is, negative definite. In accordance with the definition of Lyapunov function [42], $V(w[z_i])$ is a Lyapunov function of $w[z_i]$. Furthermore, since it is positive definite and its time derivative is negative definite, $w[z_i]$ is stable in the sense of Lyapunov stability. Hence, as $w[z_i]$ approaches to $w^*[z_i]$, $r[z_i]$ also converges to $r^*[z_i]$ since $V(w^*[z_i]) = 0$. ■

Theorem 4.2 *(Rate of Convergence) For $\forall z_i \in \mathcal{M} \backslash \{z_1\}$, the sequence of $r(t)[z_i]$ converges to $r^*[z_i]$ with the rate of $\rho \in [0, 1)$.*

Proof. To begin with, we denote the backoff window size at t as w_t. The rate of convergence [43] is defined as:

$$\lim_{t \to \infty} \frac{|\mathsf{H}(r_{t+1}[z_i])|}{|\mathsf{H}(r_t[z_i])|} = \rho \in [0, 1), \tag{4.12}$$

where $\mathsf{H}(r_t[z_i]) \neq 0$ and r_t is the data rate at t. We prove that the convergence rate of the ACWC algorithm is $\rho \in [0, 1)$ by contradiction. Assume that $\rho > 1$. Since $\rho > 1$, $|\mathsf{H}(r_{t+1}[z_i])| > |\mathsf{H}(r_t[z_i])|$ must hold. According to Equation (4.1) we consider two cases.

Case 1. $\mathsf{H}(r_t[z_i]) > 0$: When $\mathsf{H}(r_t[z_i] > 0$, $r_t[z_i]$ converges asymptotically to $r^*[z_i]$ by *Theorem* 4.1 since $r_t[z_i]$ decreases while $w_t[z_i]$ increases. Furthermore, $w_t[z_i] < w_{t+1}[z_i]$. This implies that as t increases, $\mathsf{H}(r_t[z_i]) > \mathsf{H}(r_{t+1}[z_i])$, which is contradiction.

Case 2. $\mathsf{H}(r_t[z_i]) < 0$: When $\mathsf{H}(r_t[z_i] < 0$, $r_t[z_i]$ converges asymptotically to $r^*[z_i]$ by *Theorem* 4.1 since $r_t[z_i]$ increases while $w_t[z_i]$ decreases. Furthermore, $w_t[z_i] > w_{t+1}[z_i]$. This implies that as t increases, $\mathsf{H}(r_{t+1}[z_i]) < \mathsf{H}(r_t[z_i])$, which is contradiction.

Since these two cases contradict the assumption, the sequence, $r_t[z_i]$ converges to $r^*[z_i]$ with the rate of $\rho \in [0, 1)$ as $t \to \infty$. ■

4.5.2 Analysis of VCAC Algorithm

Proposition 4.1 *(Virtual Collision Prediction) In the VCD, two instances, v_p and v_q for $p < q$ do not incur a virtual collision if and only if*

$$VC = \frac{t_{bo}[v_p] + d[v_p]}{t_{bo}[v_q]} < 1, \ (v_p, v_q \in \mathcal{V}), \tag{4.13}$$

where VC is the virtual collision factor.

Proof. Only if: We show that v_p and v_q cause a virtual collision if $VC > 1$. Since $VC > 1$, $t_{\mathrm{bo}}[v_p] + d[v_p] > t_{\mathrm{bo}}[v_q]$, which implies that during $d[v_p]$, v_q may attempt to sense the channel availability such that it has no opportunity to transmit and must backoff again since v_p has already accessed the channel prior to v_q. Thus, v_q causes a virtual collision.

If: The proof is by contradiction. Assume that $VC < 1$, yet v_p and v_q incur a virtual collision. Since v_p and v_q cause a virtual collision, either $t_{\mathrm{bo}}[v_p] = t_{\mathrm{bo}}[v_q]$ or $t_{\mathrm{bo}}[v_p] + d[v_p] > t_{\mathrm{bo}}[v_q]$

holds. The first condition indicates v_p and v_q attempt to sense the channel availability simultaneously, which implies that they incur a virtual collision, i.e., $VC > 1$. Clearly, the second condition means $VC > 1$. All these cases indicate that $VC > 1$, which contradicts the assumption that $VC < 1$. ■

Proposition 4.2 *(EBTF Schedule) The VCAC produces a schedule in order of the EBTF.*

Proof. We use induction to show that the VCAC produces a EBTF schedule. Let $\mathcal{V}$ be a schedule, i.e., a set of instances and let l be the cardinality of $\mathcal{V}$.

Induction base : Clearly the empty set $\emptyset$ is included in $\mathcal{V}$.

Induction hypothesis: Assume that, after Algorithm 8 runs, the set of instances so far added are ordered in the EBTF.

Induction step: We need to show that the set $\mathcal{V} \cup \{v_{l+1}\}$ is a EBTF schedule, where v_{l+1} is a new instance. Since Algorithm 8 adds it to $\mathcal{V}$ in order of the EBTF, the set $\mathcal{V} \cup \{v_{l+1}\}$ is also a EBTF schedule. Therefore, the theorem follows. ■

Theorem 4.3 *(Guaranteed Scheduling) Regardless of synchronous or asynchronous arrivals of instances in the VCD, the VCAC guarantees a virtual collision-free schedule.*

Proof. For two instances, v_p and v_q, we consider three cases.

Case 1. $w[v_p] = w[v_q]$ and $VC \geq 1$: This is the case where v_p and v_q arrive simultaneously and cause virtual collisions. Since $VC > 1$, their backoff times are reallocated by Algorithm 11 to prevent virtual collisions.

Case 2. $w[v_p] \neq w[v_q]$ and $VC \geq 1$: This is the case where v_p and v_q arrive asynchronously and cause virtual collisions. Like the case 1, the VCAC predicts virtual collisions by Algorithm 10 and reallocates their backoff times.

Case 3. $w[v_p] \neq w[v_q]$ and $VC < 1$: This is the case where v_p and v_q arrive asynchronously, yet they do not cause virtual collisions. Since the VCAC produces a EBTF schedule by *Proposition* 4.2, asynchronous arrivals of new instances do not cause virtual collisions.

In the three cases, the VCAC produces a virtual collision-free schedule. Therefore, theorem follows. ■

Theorem 4.4 *(Feasibility) $\mathcal{V}$ is feasible if and only if*

$$U_{BT} = \frac{\sum_{v_j \in \mathcal{V}} d[v_j]}{t_{bo}[v_k]} < 1, \ (\forall j \leq k \leq |\mathcal{V}|). \tag{4.14}$$

where U_{BT} is the utilization factor of backoff times.

Proof. Only if: Without loss of generality, the instances in $\mathcal{V}$ are ordered by nondecreasing $t_{\rm bo}[v_j]$ so that v_1 has the smallest $t_{\rm bo}$ and v_l has the largest $t_{\rm bo}$. We show that $\mathcal{V}$ is not feasible if $U_{\rm BT} > 1$. In equation (4.14), the numerator is the sum of $t_{\rm bo}[v_j]$ for $\forall v_j \in \mathcal{V}$. If $U_{\rm BT} > 1$ (which implies that the numerator exceeds the upper bound, $t_{\rm bo}[v_k]$), then at least one instance causes a virtual collision with any other instance so that $\mathcal{V}$ is not feasible.

If: We show the sufficiency by contradiction. Assume that equation 4.14 holds, yet $\mathcal{V}$ is not feasible. Since $\mathcal{V}$ is not feasible, $\mathcal{V}$ has at least one v_j, whose $t_{\rm bo}[v_j] - t_{\rm bo}[v_{j-1}] < d[v_{j-1}]$. Then, the sum of $t_{\rm bo}[v_j]$ for $\forall j \leq k \leq l$ is greater than $t_{\rm bo}[v_k]$ since $t_{\rm bo}[v_j] + d[v_j] > t_{\rm bo}[v_k]$. That is, $\sum_{v_j \in \mathcal{V}} t_{\rm bo}[v_j] + d[v_j] > t_{\rm bo}[v_k]$ for $\forall j \leq k \leq l$, which contradicts the assumption that $U_{\rm BT} < 1$. ■

Proposition 4.3 *(Handling Overload Conditions) On the overload condition, the VCAC replaces v_p with v_{new} if $c_t[v_p]$ is the smallest and satisfies $c_t[v_p] < c_t[v_{new}]$, where v_{new} is a new instance and $\mathsf{C}(t)$ is the cost value at t.*

Proof. To handle the overload condition, i.e., $U_{\rm BT} \geq 1$, the VCAC resorts to Algorithm 9. When adding $v_{\rm new}$ violates the feasibility condition, Algorithm 9 finds out the instance v_p

with the smallest $\mathsf{C}(t)$ in the schedule and checks if $c_t[v_p] < c_t[v_{\text{new}}]$ for $v_p \in \mathcal{V}$ and $v_{\text{new}} \notin \mathcal{V}$ (See lines 4-10 in Algorithm 9). Then, it replaces v_p with v_{new} as long as v_{new} does not violate the feasibility condition (See lines 11-18 in Algorithm 9). ■

Theorem 4.5 *(Algorithm Complexity) The algorithm complexity of the VCAC is $O(l)$, where $l = |\mathcal{V}|$.*

Proof. The algorithms that have an impact on the algorithm complexity of the VCAC are Algorithm 9, Algorithm 10 and Algorithm 11. We consider the instructions executed for the value of l to be the basic operations. In Algorithm 10, the number of instructions executed in the **for** loop and two if statements is $l+2 \in O(l)$. In Algorithm 11, the number of instructions executed in the **for** loop is $l-1 \in O(l)$. In Algorithm 9, the number of instructions executed in the **for** loop and if statement is $l + 2 + (l + 2) + (l - 1) = 3l + 3 \in O(l)$. Therefore, the algorithm complexity of the VCAC is $(l+2)+(l-1)+(3l+3) = 5l+4 \in O(l)$. The theorem follows. ■

4.6 Implementation Considerations

The MSD model is easily integrated into the IEEE 802.15.4 MAC layer. To realize it at the MAC layer, four components must be implemented (See Figure 4.3).

Firstly, the session administrator is responsible for creating a new session and maintaining active sessions. In conjunction with the NSE component, it notifies individual VMAC entities that they need not only to update their achieved transmission rate, but also to calculate their contention window size for next transmission round.

Secondly, the packet classifier relays the packets from the upper layer to the VMAC entities.

Basically, the packets are classified by the cross-layered packet classification. This classification method utilizes a pair of source and destination IP addresses in conjunction with a pair of source and destination ports and puts them into the corresponding VMAC entities.

Thirdly, the NSE component is responsible for estimating the network state and providing the achieved data rate. Since this component is coupled with the PHY layer, it utilizes the channel state information provided by the PHY layer.

Lastly, the VMAC entities implement the ABWC and VCAC algorithms. The ABWC updates the achieved transmission rate and contention window size every transmission, and the VCAC works whenever the VMAC entities instantiate new packets for backoff. Note that a VMAC entity is allowed to instantiate a new packet only when its ongoing instance is finished.

4.7 Simulation Study I

In this section, we conduct simulation-based experiments to validate our analysis and to evaluate the performance of the ABWC and VCAC algorithms using the ns-2 simulator [29].

4.7.1 Simulation Settings

For the simulation, we deploy five wireless sensors in a 50×50 area and generate Constant Bit Rate (CBR) traffic for three data flows, f_1, f_2 and f_3. The data frame length is 63 bytes and each frame contains PHY_HDR, MAC_HDR and $PAYLOAD$. For simplicity, we denote a data flow k as $f_k(r^* : \mathrm{WS}_s, \mathrm{WS}_d)$ where r^* is the rate requirement in kbps and WS_s and WS_d are a source node and a destination node respectively. Figure 4.8 shows the geographical deployment of five wireless sensors and three data flows.

Wireless sensor i	Location in 50x50 area
WS0	(12,28)
WS1	(15,15)
WS2	(27,20)
WS3	(35,9)
WS4	(48,14)

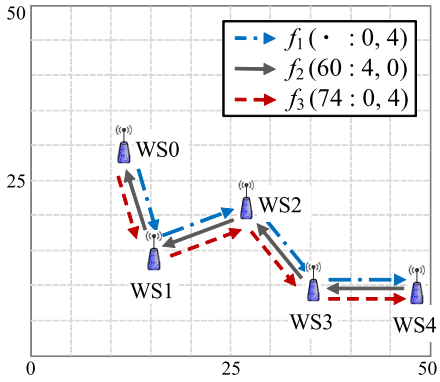


Figure 4.8: Deployment of wireless sensors and three data flows.

Table 4.1 shows initial settings of three data flows. Note that the required data rate of f_1 is not given since it is a best-effort flow. The data rate at the MAC layer is chosen as 250 kbps and all the other parameters are chosen as defaults defined in the IEEE 802.15.4 standard. The simulation duration is given by 100 seconds.

Table 4.1: Initial settings of three data flows

Data flow	Forwarding Path	w	$(\gamma_{\text{bw}}, \gamma_{\text{cli}}, \gamma_{\text{nci}})$	N_{sampl}
$f_1(\cdot : 0, 4)$	$01, 2, 3, 4$	32	(2.5E-02,3.2E-01,8.0E-01)	25
$f_2(60 : 4, 0)$	$4, 3, 2, 1, 0$	32	(2.5E-02,3.2E-01,8.0E-01)	25
$f_3(74 : 0, 4)$	$0, 1, 2, 3, 4$	32	(2.5E-02,3.2E-01,8.0E-01)	25

4.7.2 Performance Evaluation

Figures 4.9 and 4.10 show the achieved data rate and the backoff window size achieved by $f_1(\cdot)$ where $\cdot$ indicates the best-effort flow. We observe that the achieved data rate and backoff window size of f_1 fluctuate drastically as time increases. This is because its achieved data rate and backoff window size depend on both f_2 and f_3's backoff window sizes in accordance with Equation (4.1).

Note that at $t = 0$, the highest data rate is fulfilled during the simulation since Address Res-

olution Protocol (ARP) [44] and Ad hoc On-demand Distance Vector (AODV) [45] packets with the highest priority are sent at that time. It is worth noting that f_1 maintains lower data rate than f_2 and f_3, as compared to Figures 4.11 and 4.13. We also observe that the data rate is inversely proportional to the backoff window size as shown in all the figures.

Figures 4.11 and 4.12 show the achieved data rate and the backoff window size achieved by $f_2(60)$, where 60 is the required data rate. We observe in Figure 4.11 that the data rate achieved by each of the four wireless sensors converges asymptotically to the required data rate, i.e., 60 kbps. This validates *Theorem* 4.1.

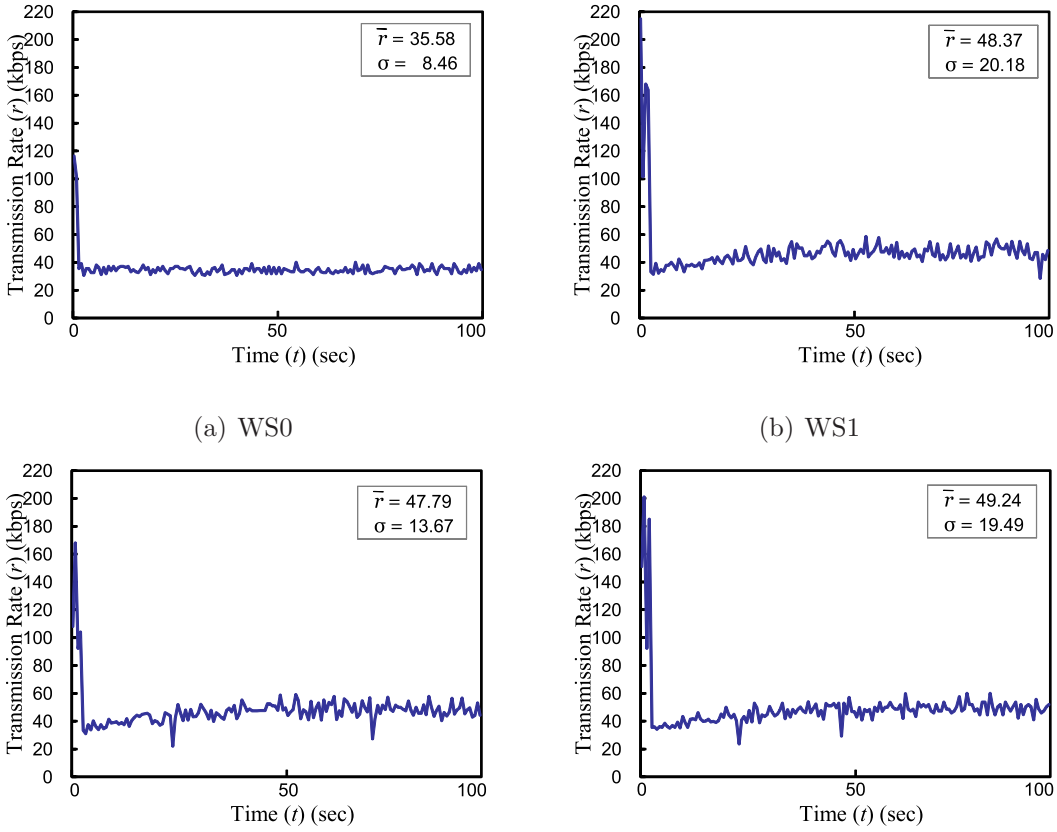
Figure 4.11(a) shows that the data rate drastically drops down and go up to $f_2(60)$ at t_0, t_1 and t_2. Such a sharp decrease in the data rate is due to increased F_{nse} caused by collisions in the LCD, which leads to increasing the backoff window size as shown in Figure 4.12(a) (See Equation (4.1)).

Figures 4.13 and 4.14 show the data rate and backoff window size achieved by f_3 established for the rate-sensitive flow, $f_3(74)$. Like f_2's data rate, f_3's data rate achieved by four wireless sensors converges asymptotically to the required data rate, $f_3(74)$ kbps as shown in Figure 4.13. This also validates *Theorem* 4.1.

We observe that the number of local collisions at WS0 and WS2 is greater than those at WS1 and WS3 since drastic drops in the data rate indicate that local collisions have occurred as shown in Figure 4.13. In particular, each backoff window size in Figure 4.14 makes gently sloping curves from $t = 0$ and maintains a fixed value. This is because the data rate converges asymptotically to the required data rate (See Equation (4.1)).

4.7.3 Rate of Convergence

In this section, we evaluate convergence rates of both f_2 and f_3. Figures 4.15 and 4.16 show the convergence rate achieved by f_2 and f_3 respectively. In Figure 4.15(a), WS4's



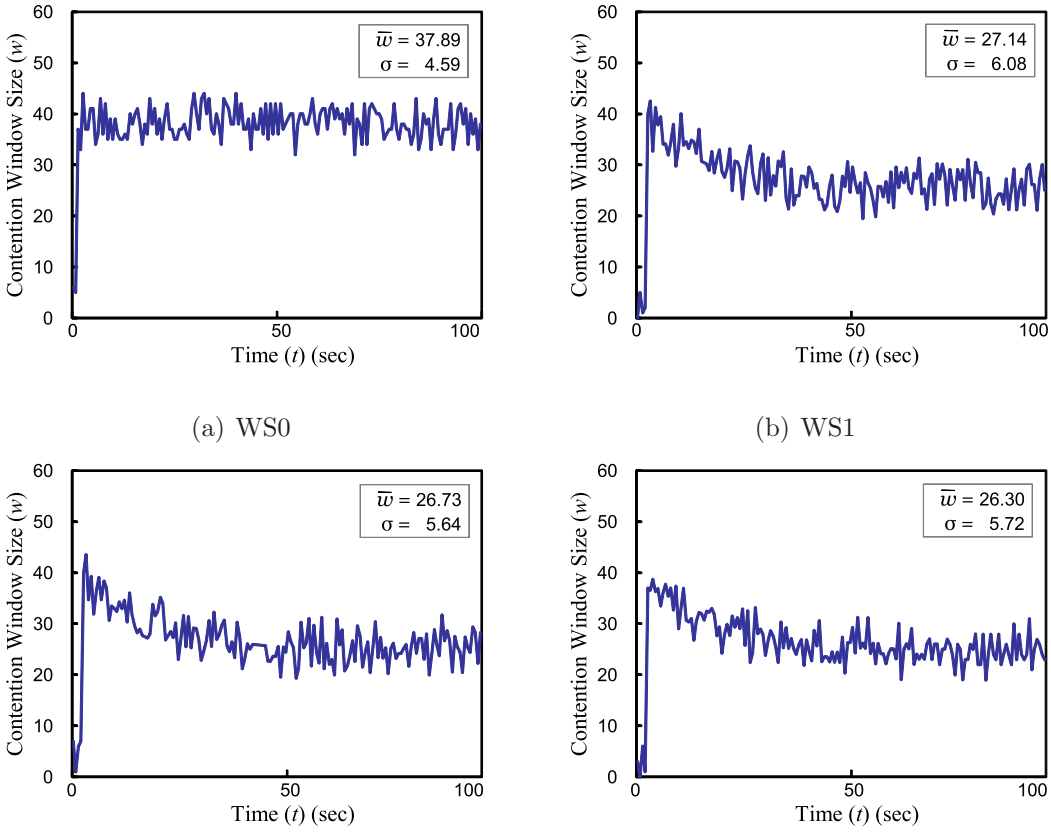
(a) WS0

(b) WS1

(c) WS2

(d) WS3

Figure 4.9: Achieved data rate of $f_1(\cdot : 0, 4)$. $\bar{r}$ is the averaged date rate. σ is the standard deviation of the achieved data rate.



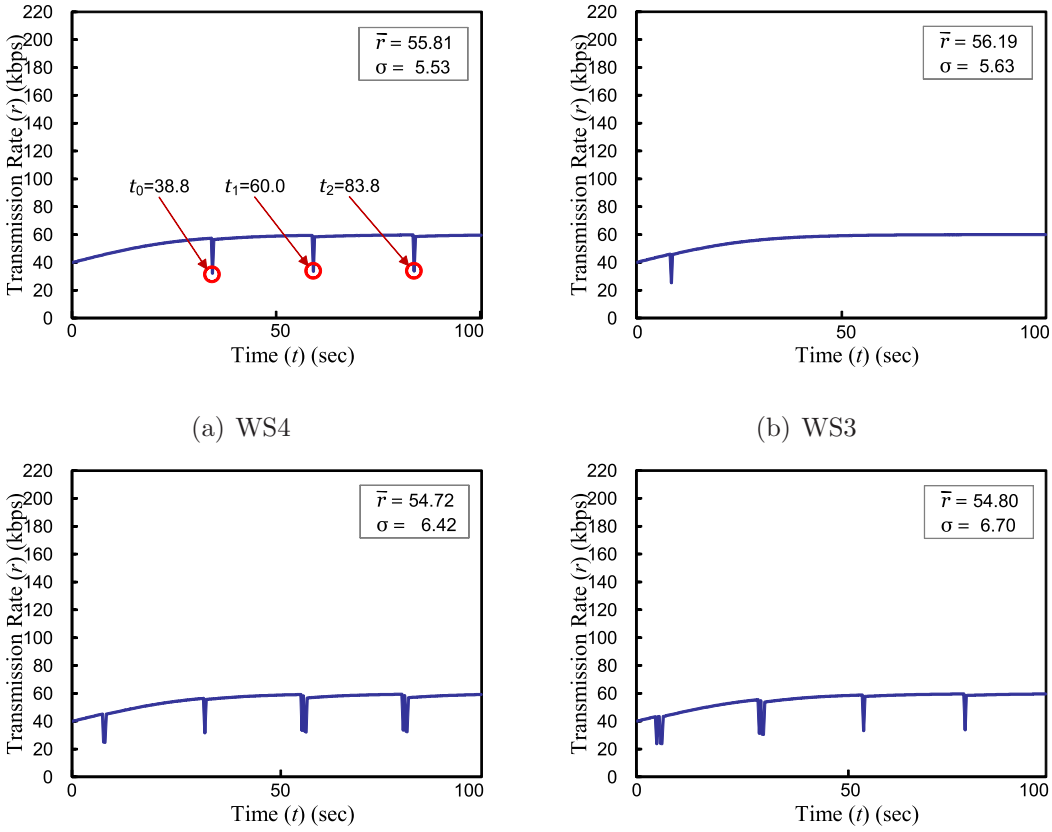
(a) WS0

(b) WS1

(c) WS2

(d) WS3

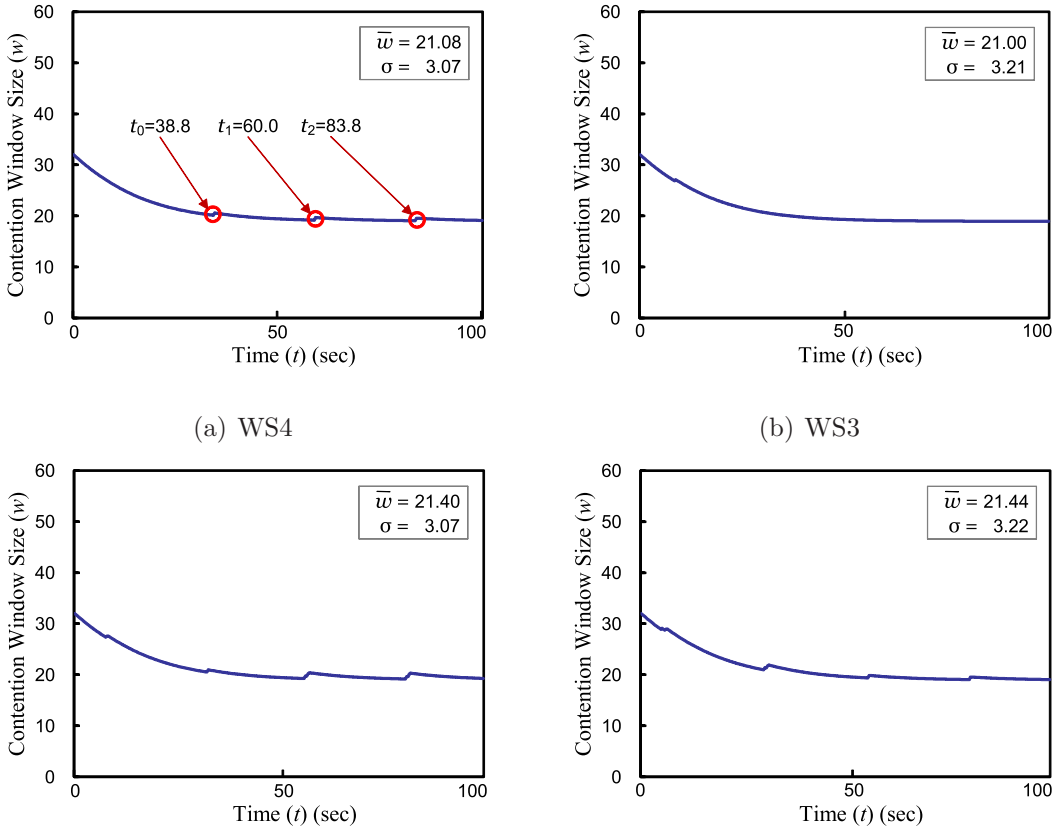
Figure 4.10: Backoff window size of $f_1(\cdot : 0, 4)$. $\bar{w}$ is the averaged backoff window size. σ is the standard deviation of the achieved backoff window size.



(a) WS4 (b) WS3

(c) WS2 (d) WS1

Figure 4.11: Achieved data rate of $f_2(60:4,0)$. $\bar{r}$ is the averaged date rate. σ is the standard deviation of the achieved data rate.



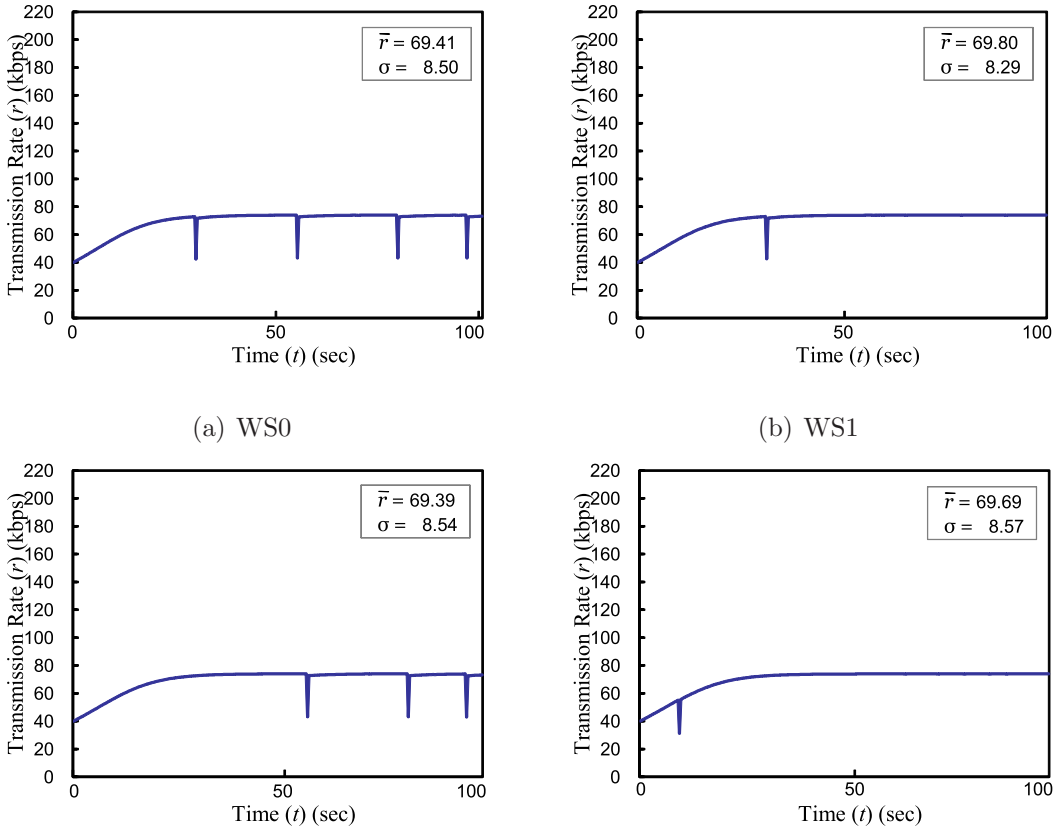
(a) WS4

(b) WS3

(c) WS2

(d) WS1

Figure 4.12: Backoff window size of $f_2(60:4,0)$. $\bar{w}$ is the averaged backoff window size. σ is the standard deviation of the achieved backoff window size.



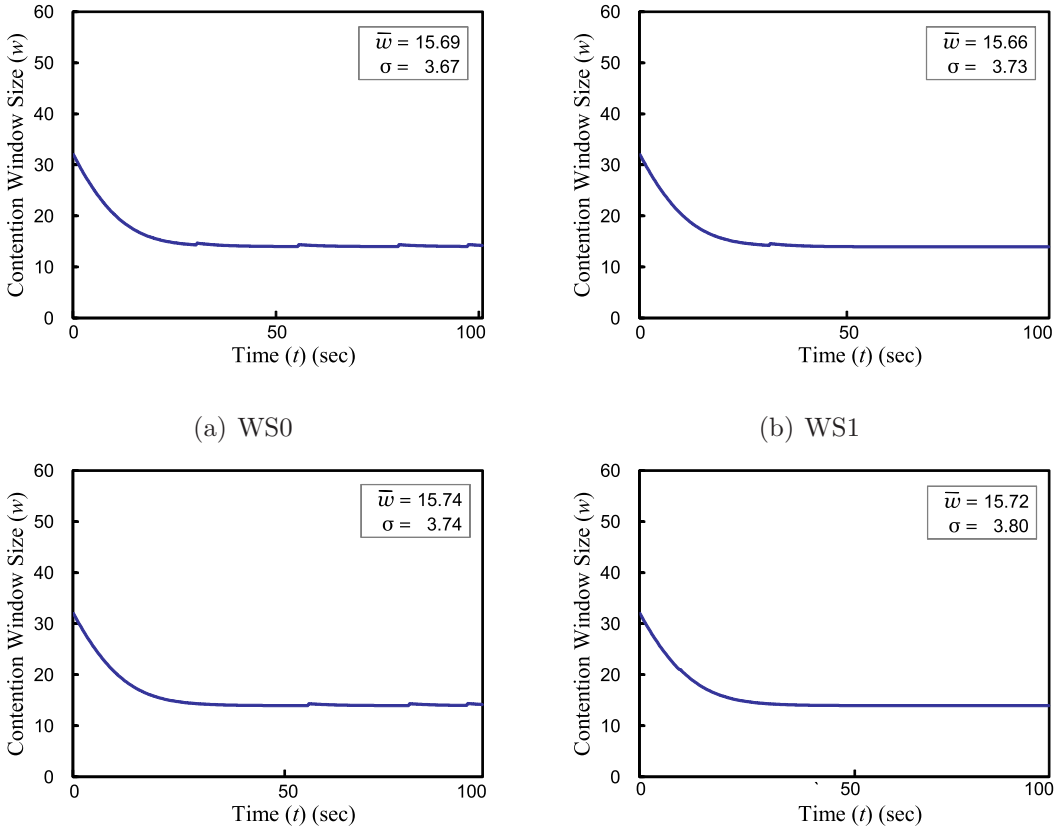
(a) WS0

(b) WS1

(c) WS2

(d) WS3

Figure 4.13: Achieved data rate of $f_3(74 : 0, 4)$. $\bar{r}$ is the averaged date rate. σ is the standard deviation of the achieved data rate.



(a) WS0 (b) WS1

(c) WS2 (d) WS3

Figure 4.14: Backoff window size of $f_3(74 : 0, 4)$. $\bar{w}$ is the averaged backoff window size. σ is the standard deviation of the achieved backoff window size.

convergence rate maintains less than 1 during the simulation time, while it drops down and goes up drastically at t_0, t_1 and t_2. As in Figure 4.11(a), such a significant fluctuation happens only when a wireless sensor collides with its neighbors in the LCD, leading to increasing its backoff window size.

It is worth noting that WS4's convergence rate gradually rises right after such a local collision. This implies that once the convergence rate starts falling down, then it keeps going down and maintains less than 1 until it rises again due to local collisions. Likewise, WS2's convergence rate in Figure 4.15 and both WS1 and WS2's convergence rate in Figure 4.16 go down gradually. In particular, in Figures 4.16(b) and 4.15(d), both WS1 and WS3's convergence rates reach 0, which means that their data rates converge exactly to $f_3(74)$. This validates *Theorem* 4.2.

We denote the number of updates of w achieved to converge to a required data rate as N_w. Figure 4.17 shows the N_w of f_1 and f_2 depending on $z_i[\theta]$ and γ_{cw} defined in Table 4.2 (See Equation (4.9)). We assume that f_2 and f_3 are served by z_2 and z_3 respectively at all wireless sensors. We observe that the larger $z_i[\theta]$, the smaller N_2.

It is worth noting that a large scaling factor not only reduces N_w performed for convergence to the required data rate, but also makes convergence much faster. However, too large a scaling factor does not always guarantee accurate convergence. In Figure 4.17(a), the N_w of WS1 and WS4 goes up at $\gamma_{\text{cw}} = 9$ and $\gamma_{\text{cw}} = 11$. This is because the N_w of those wireless sensors has suffered from the backoffs caused by local collisions or the VCAC algorithm (See Equations (4.1), (4.4) and (4.5), and lines 4-6 in Algorithm 7).

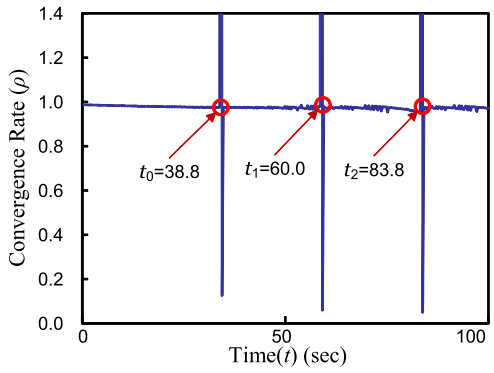
4.7.4 Comparison of VCH and VCAC

In this section we compare two algorithms, Virtual Collision Handler (VCH) [11] and VCAC in terms of latency. To evaluate them, we employ two wireless sensors, WS0 and WS1, and

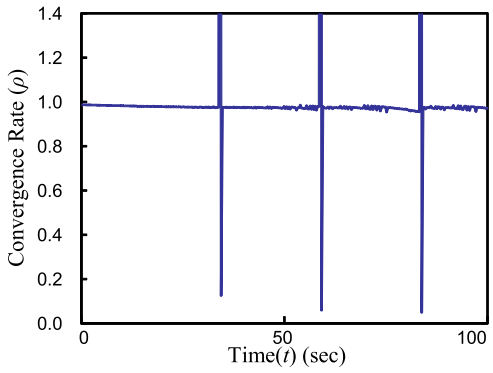
Table 4.2: $z_i[\theta]$ and γ_{cw}.

$\gamma_{cw} \times 10^{-2}$	4	5	6	7	8
$z_2[\theta]$ of f_2	1.79E-02	2.24E-02	2.69E-02	3.13E-02	3.58E-02
$z_3[\theta]$ of f_3	2.21E-02	2.76E-02	3.31E-02	3.87E-02	4.42E-02

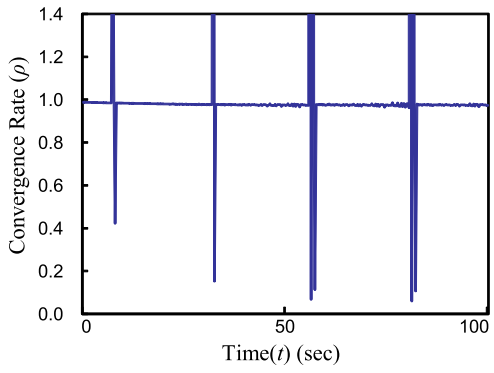
$\gamma_{cw} \times 10^{-2}$	9	10	11	12	13
$z_2[\theta]$ of f_2	4.03E-02	4.48E-02	4.93E-02	5.37E-02	5.82E-02
$z_3[\theta]$ of f_3	4.97E-02	5.52E-02	6.07E-02	6.63E-02	7.18E-02



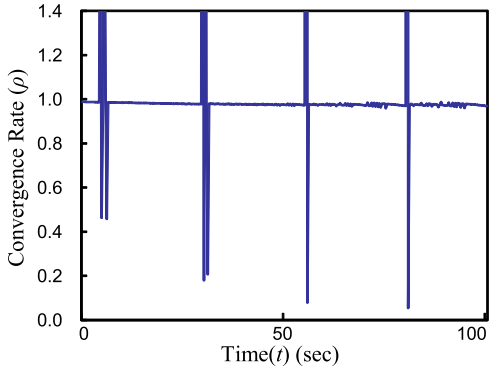
(a) WS4



(b) WS3

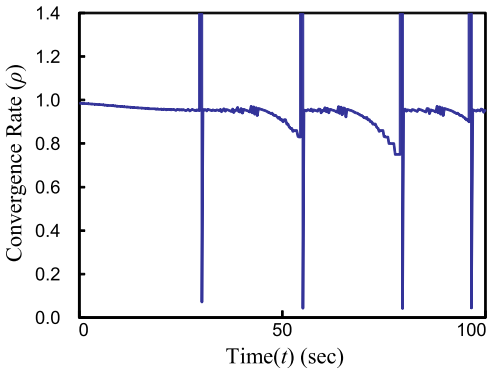


(c) WS2

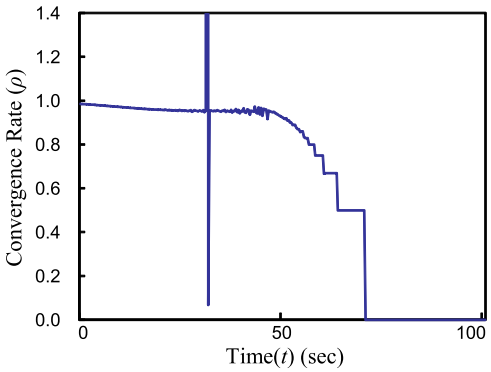


(d) WS1

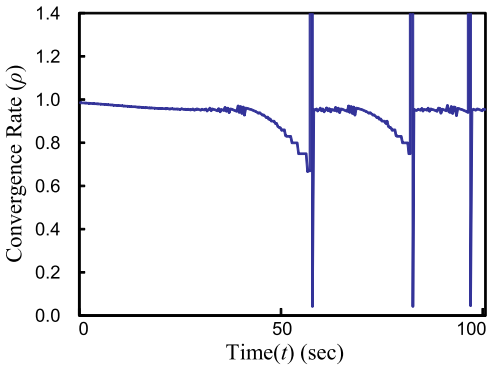
Figure 4.15: Convergence rate of $f_2(60 : 4, 0)$.



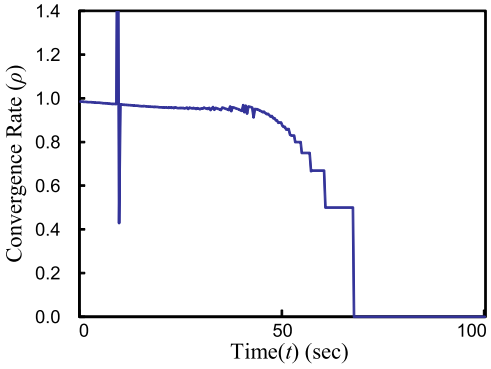
(a) WS0



(b) WS1

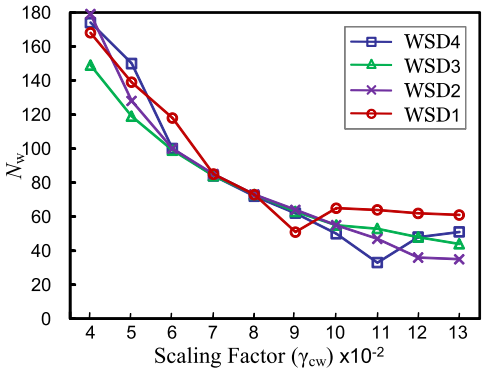


(c) WS2

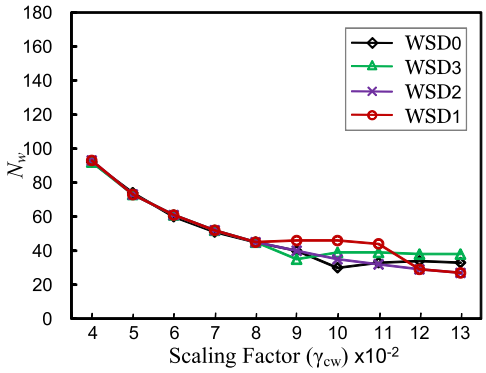


(d) WS3

Figure 4.16: Convergence rate of $f_3(74:0,4)$.



(a) $f_2(60:4,0)$



(b) $f_3(74:0,4)$

Figure 4.17: N_w performed for convergence to $f_2(60)$ and $f_3(74)$.

four data flows, $f_1(\cdot : 0, 1)$, $f_2(30 : 0, 1)$, $f_3(32 : 0, 1)$ and $f_4(34 : 0, 1)$. The data frame length is 63 bytes and each frame contains PHY_HDR, MAC_HDR and $PAYLOAD$. For simplicity, we assume that four VMACs, z_1, z_2, z_3 and z_4 on WS0 serve f_1, f_2, f_3 and f_4 respectively, and data frames on these VMACs are backlogged. Figure 4.18 shows these four data flows from WS0 to WS1.

Figure 4.19 shows the latency spent on those four VMACs. We observe that Δt_{ta} achieved by the VCH is longer than those achieved by the VCAC. This is because the VCH chooses one of the instances colliding with one another in the VCD, allows it to be sent and has the remaining colliding instances backoff again. As a result, these re-backoffs lead to longer latency since the number of backoffs increase.

Figure 4.20 shows that the number of backoffs by the VCH is greater than those by the VCAC. This implies that the VCH fails to handle virtual collisions so that it leads to increase in the latency while the VCAC produces the virtual collision-free schedule so that it does not increase the latency (See *Theorem* 4.4). This validates *Proposition* 4.1 and *Theorem* 4.3.

Figure 4.19(a) shows that Δt_{ta} of z_1 achieved by VCAC is more fluctuated than those achieved by the others. This is because z_1 serves the best-effort flow and has the lowest priority such that it backoffs again whenever a virtual collision occurs.

In particular, Figures 4.19(b), 4.19(c) and 4.19(d) show uneven patterns of Δt_{ta} achieved by the VCAC. This indicates that due to virtual collisions, shorter backoff window sizes are reallocated by the VCAC (See Algorithm 11). However, this temporary fluctuation in Δt_{ta} is adjusted by the ABWC such that Δt_{ta} maintains a constant latency again.

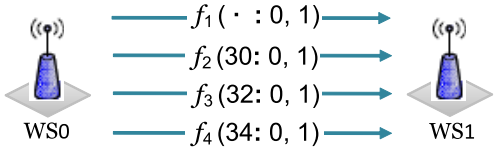
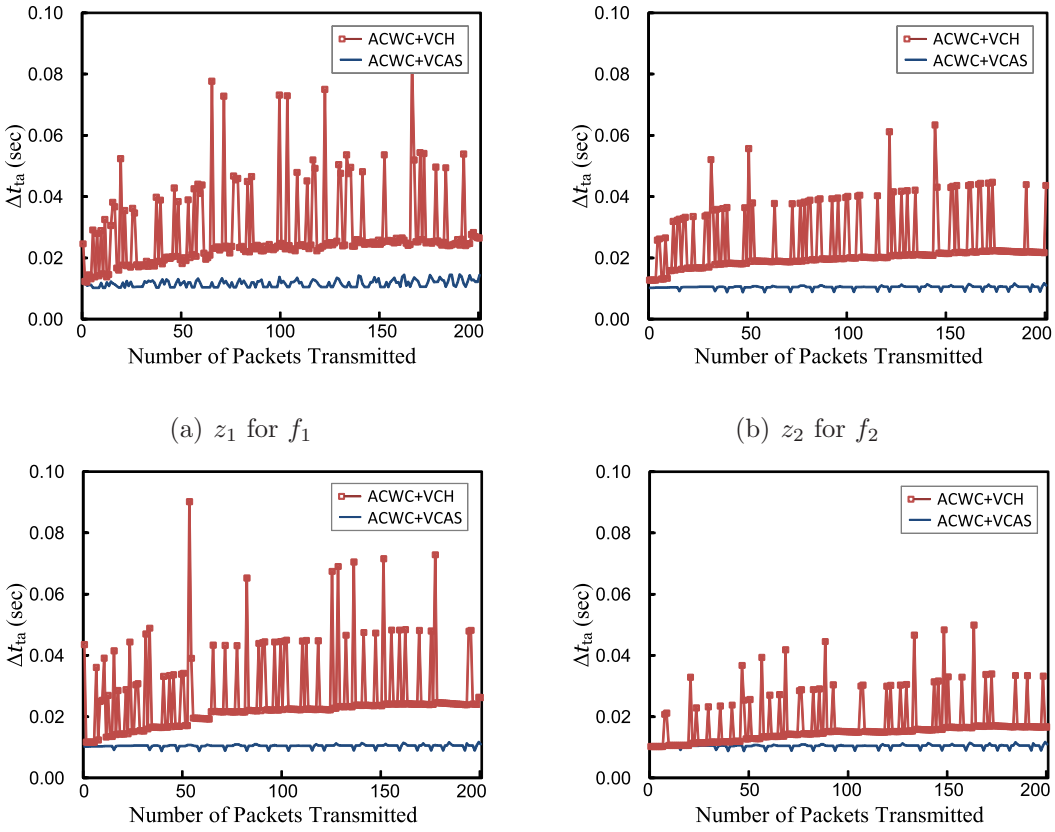


Figure 4.18: Four data flows from WS0 to WS1.



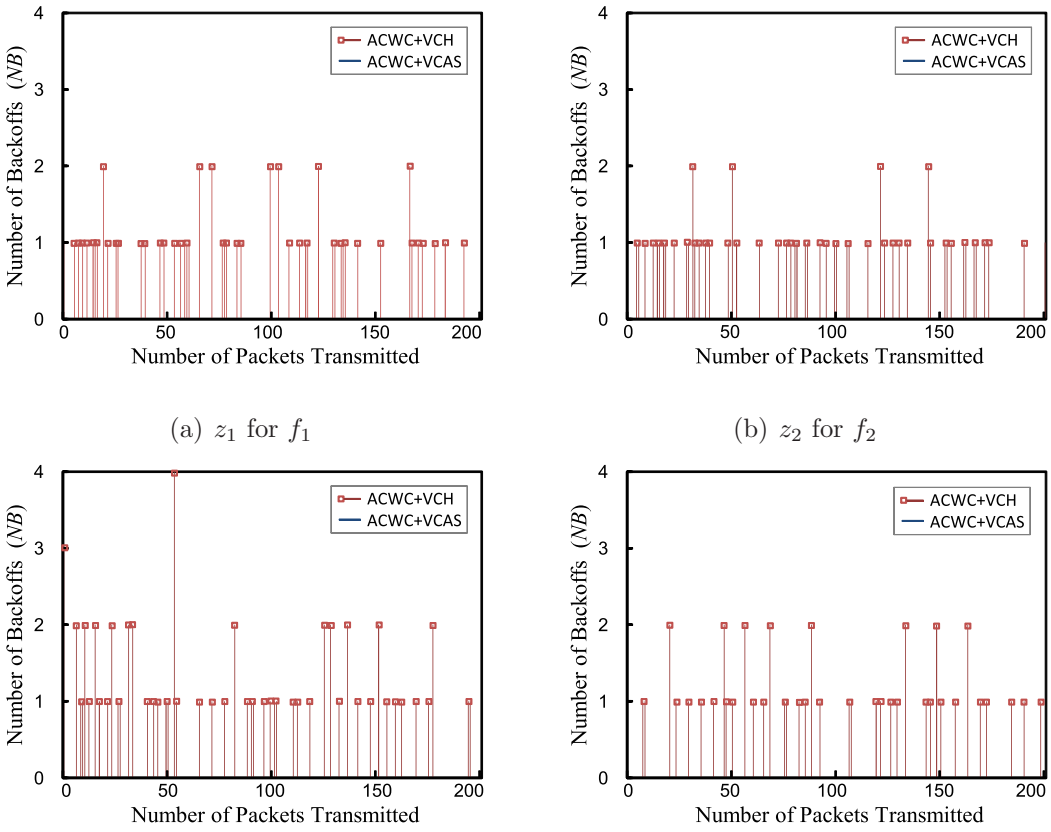
(a) z_1 for f_1

(b) z_2 for f_2

(c) z_3 for f_3

(d) z_4 for f_4

Figure 4.19: Latency spent at four VMACs, z_1, and z_2, z_3 and z_4.



(a) z_1 for f_1

(b) z_2 for f_2

(c) z_3 for f_3

(d) z_4 for f_4

Figure 4.20: Number of backoffs at four VMACs, z_1, and z_2, z_3 and z_4.

4.8 Simulation Study II

In this section, we conduct simulation-based experiments to compare the performance of the MSD and existing approaches using the ns-2 simulator [29].

4.8.1 Simulation Settings

For the simulation, we deploy twelve wireless sensors in a 100×100 area and generate Constant Bit Rate (CBR) traffic for four data flows, f_1, f_2, f_3 and f_4 whose the frame length is 63 bytes and a data frame contains PHY_HDR, MAC_HDR and $PAYLOAD$. Table 4.3, Figure 4.21 and Table 4.4 show the geographical deployment of twelve wireless sensors and the initial settings of four data flows.

Table 4.3: Deployment of twelve wireless sensors

Wireless sensor i	Location in 100×100 area
WS0	(10,90)
WS1	(25,90)
WS2	(35,80)
WS3	(50,70)
WS4	(50,50)
WS5	(95,70)
WS6	(80,80)
WS7	(65,80)
WS8	(60,40)
WS9	(60,25)
WS10	(50,10)
WS11	(35,5)

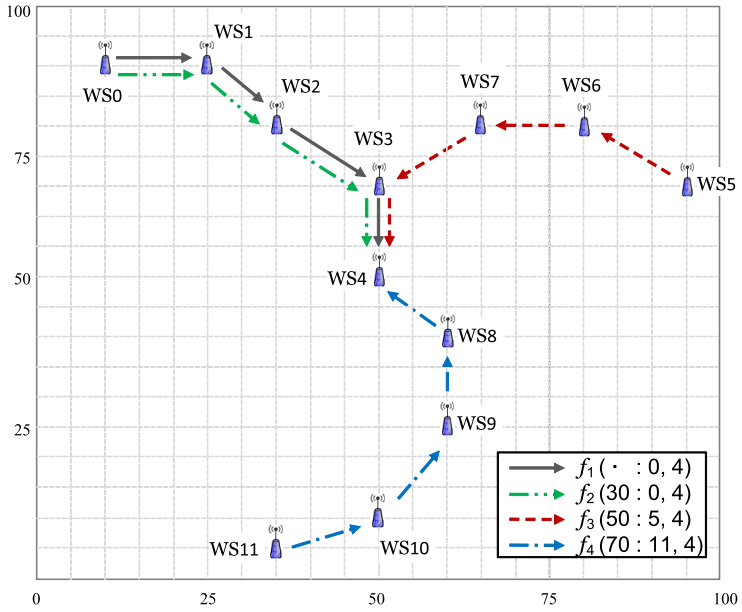


Figure 4.21: Four data flows.

Table 4.4: Initial settings of four data flows

Data flow	Forwarding Path	w	$(\gamma_{\text{bw}}, \gamma_{\text{cli}}, \gamma_{\text{nci}})$	N_{sampl}
$f_1(\cdot : 0, 4)$	$0, 1, 2, 3, 4$	64	(2.5E-02,3.2E-01,8.0E-01)	25
$f_2(30 : 0, 4)$	$0, 1, 2, 3, 4$	64	(2.5E-02,3.2E-01,8.0E-01)	25
$f_3(50 : 5, 4)$	$5, 6, 7, 3, 4$	64	(2.5E-02,3.2E-01,8.0E-01)	25
$f_4(70 : 11, 4)$	$11, 10, 9, 8, 4$	64	(2.5E-02,3.2E-01,8.0E-01)	25

Table 4.5: Initial settings of PSD-S and PSD-M

	Backoff Exponent (BE)			
	f_1	f_2	f_3	f_4
PSD-S	4	2	3	2
PSD-M	[2,5]	[0,5]	[0,5]	[0,5]

4.8.2 Performance Evaluation

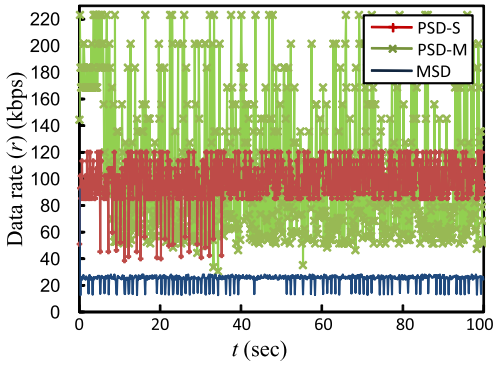
In terms of convergence to the rate requirement, we compare three algorithms: 1) multi-level service differentiation (PSD-S) with a single FIFO queue [38] [39] 2) prioritized multi-queues service differentiation (PSD-M) [40] and 3) MSD. Throughout the simulation, PSD-S's BEs are initialized to BE=2 for f_2 and f_4, BE=3 for f_3 and BE for f_1, while PSD-M's BE is initialized to [0,5] for f_2, f_3 and f_4, and [2,5] for f_1.

Figures 4.22 shows that the data rate achieved by PSD-S, PSD-M and MSD with respect to f_1. We observe that the data rates in PSD-S, PSD-M and MSD are drastically fluctuated, while the extent of MSD's fluctuation is not as much as those in PSD-S and PSD-M. It is worth nothing that the MSD's data rate is less than PSD-S and PSD-M since its backoff window always has the largest one out of active $w[z_i]$ for $\forall z_i \in \mathcal{M}$ (See equation (4.1)).

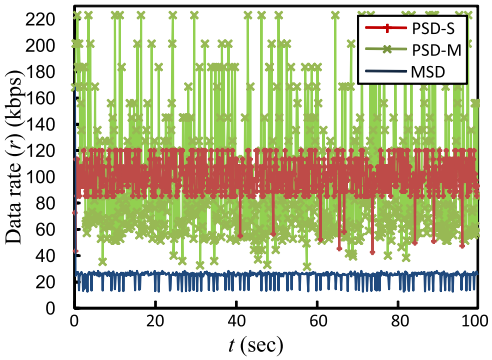
Figures 4.23, 4.24 and 4.25 show the data rates achieved by PSD-S, PSD-M and MSD with respect to f_2, f_3 and f_4. In the Figures, the data rates achieved by MSD converge to $f_2(30)$ (Figure 4.23), $f_3(50)$ (Figure 4.24), and $f_4(70)$ (Figure 4.25) respectively. This validates *Theorem* 4.1. We observe that the data rates achieved by PSD-S and PSD-M are fluctuated drastically as in figures 4.22.

In Figures 4.23 and 4.25, PSD-S's data rates are higher than those in Figures 4.22 and 4.24. This is because the backoff windows in z_2 and z_4 are less than those in z_1 and z_3. We observe in Figures 4.23(d) and 4.24(d) that the data rates in the dashed circle are higher than the others. This is because the VCA reallocates the backoff window to avoid virtual collisions in the VCD.

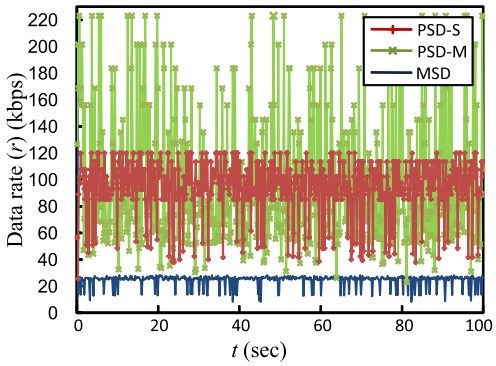
This validates *Proposition* 4.1 and *Theorem* 4.3. Figures 4.23(c), 4.23(d), 4.24(c), 4.24(d), 4.25(c) and 4.25(d) show that some of the achieved rates by the MSD do not meet the rate requirements because of the local collision and congestion caused by neighboring wireless sensors.



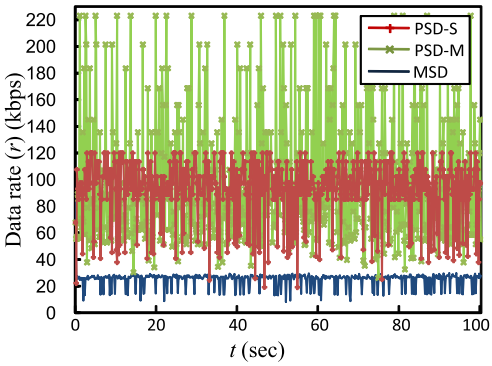
(a) $f_1(\cdot : 0, 1)$



(b) $f_1(\cdot : 1, 2)$

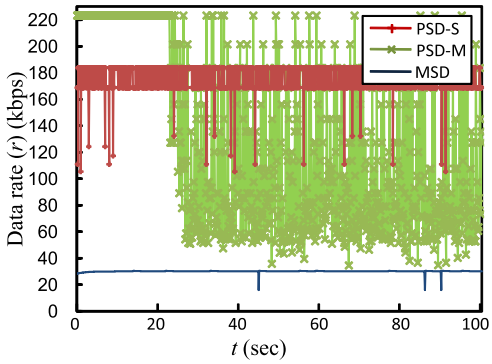


(c) $f_1(\cdot : 2, 3)$

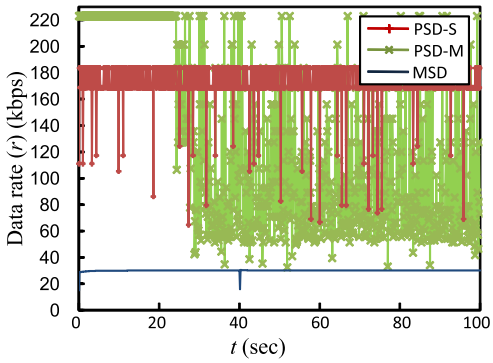


(d) $f_1(\cdot : 3, 4)$

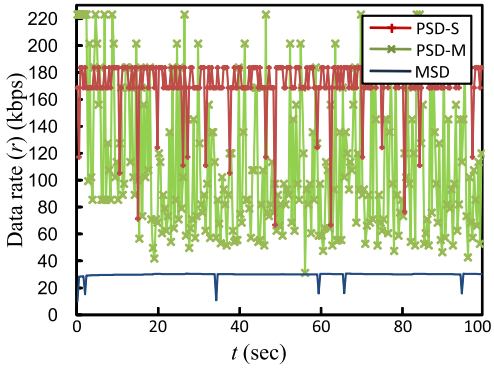
Figure 4.22: Achieved data rate of f_1.



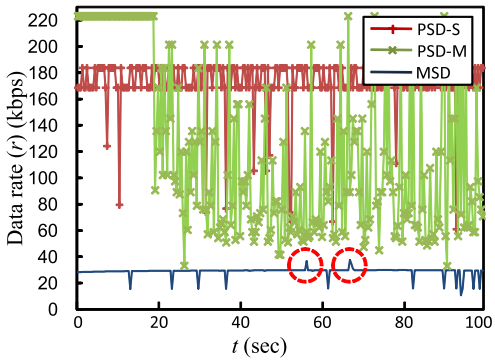
(a) $f_2(30:0,1)$



(b) $f_2(30:1,2)$

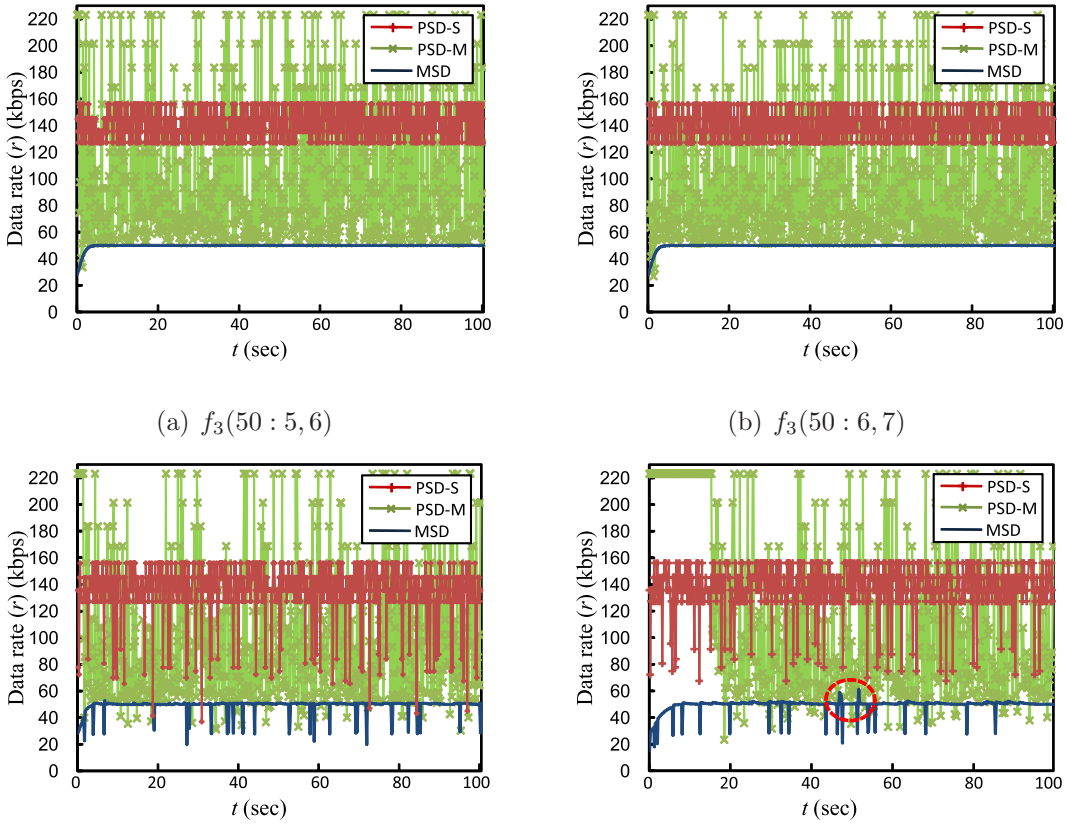


(c) $f_2(30:2,3)$



(d) $f_2(30:3,4)$

Figure 4.23: Achieved data rate of f_2.



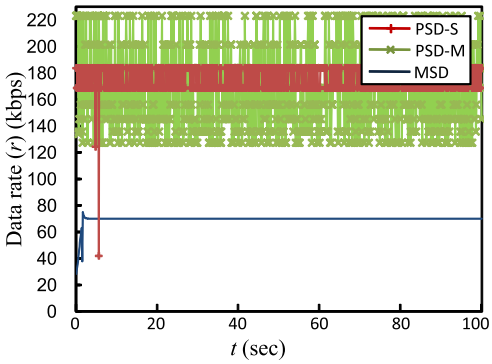
(a) $f_3(50:5,6)$

(b) $f_3(50:6,7)$

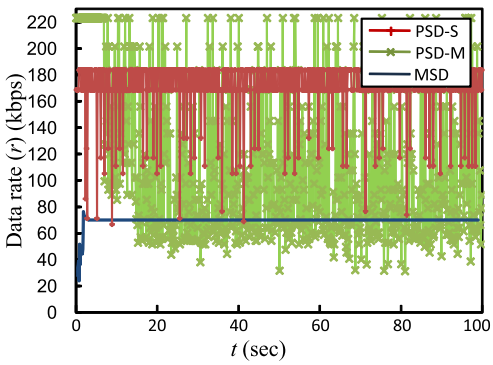
(c) $f_3(50:7,3)$

(d) $f_3(50:3,4)$

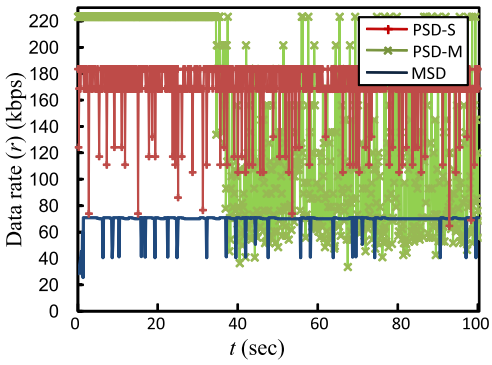
Figure 4.24: Achieved data rate of f_3.



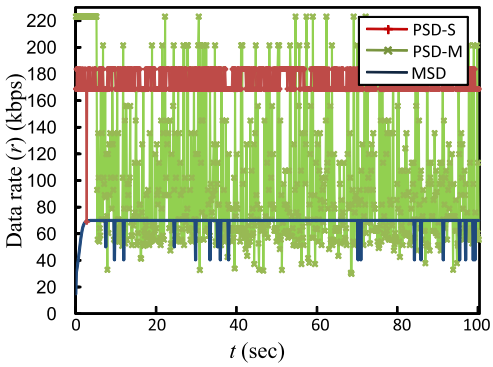
(a) $f_4(70:11,10)$



(b) $f_4(70:10,9)$



(c) $f_4(70:9,8)$



(d) $f_4(70:8,4)$

Figure 4.25: Achieved data rate of f_4.

4.9 Summary

In this chapter, we presented the MSD model operating in the nonbeacon-enabled peer-to-peer network. Unlike existing priority-based service differentiation models, the MSD defines the independent VMAC that consists of a transmission queue and the Adaptive Backoff Window Control (ABWC). The ABWC algorithm adjusts the backoff window to reflect the local network state in the local collision domain. Since the VMACs serve multiple rate-sensitive flows, it is possible that more than one data frame collide with each other when their backoff times expire simultaneously. To solve such a virtual collision in the virtual collision domain, we designed the VCAC algorithm. The VCAC algorithm prevents virtual collisions and preempts packets with the minimal cost in the virtual collision domain. By analyzing these algorithms, we proved that the ABWC component enables the achieved data rate to converge to the rate requirement and the VCAC component produces a virtual-collision-free schedule to avoid degradation of the achieved data rate. Through the simulation, we validated our analysis and show the MSD outperforms existing algorithms.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this dissertation we have addressed two challenging problems in the IEEE 802.15.4 LR-WPAN: a) optimal GTS allocation and scheduling for delay-sensitive transactions in the beacon-enabled start network and b) multirate service differentiation in the nonbeacon-enabled peer-to-peer network.

In Chapter 3 we have proposed a novel GTS allocation and scheduling algorithm, called GAS, to meet the delay constraints of delay-sensitive transactions in the IEEE 802.15.4 star topology. The GAS proved to be optimal and work-conserving so that it achieves the maximal GTS utilization and can always find a feasible schedule if there exists a feasible schedule. In addition, it smoothes out the traffic of a transaction by distributing the GTSs of a transaction over as many beacon intervals as possible while satisfying the delay constraint of the transaction.

By doing so, it reduces the average service starting time of transactions and enables concurrent services to more transactions. This can significantly benefit many delay-sensitive applications, where the starting time of the first message and the stability of traffic have great impact on the performance of these applications. Simulation results also demonstrate

that the GAS significantly outperforms both the FCFS-based scheduling algorithm and the EDF scheduling algorithm.

From a practical point of view, the GAS can be easily integrated into the IEEE 802.15.4 protocol with a minor change. It has no protocol overhead since it does not need to exchange additional control packets, as compared with other protocols. At a wireless sensor, only transaction parameters are inserted into a $GTSRequest$ message when a node has transactions to be transmitted. Since the maximum number of GTSs are up to 7, however, the amount of memory can be adjusted depending on the available resources that wireless sensors can provide. Despite the small use of memory resources, the GAS involves two implementations (feasibility examination and GTS assignment) in consuming the energy. The GTS assignment consumes more energy than the feasibility examination since the GAS runs it every beacon interval while the feasibility examination is performed only when new $GTSRequest$ messages are received.

Our extensive simulation results demonstrated that the GAS has excellent performance under bursty, periodic and aperiodic transmissions of transactions. In particular, these results indicate that the delay constraint meet ratio (DMR) of our algorithm is up to 100% higher than the FCFS-based scheduling algorithm. Our algorithm differs from the existing ones in that it is an on-line scheduling algorithm and allows transmissions of bursty and periodic messages with delay constraints even when the network is overloaded.

In Chapter 4 we addressed the multirate-based service differentiation model for rate-sensitive applications in the nonbeacon-enabled peer-to-peer network. Unlike existing priority-based service differentiation models, the MSD defines the independent Virtual Medium Access Controls (VMACs), each of which consists of a transmission queue and the Adaptive Backoff Window Control (ABWC).

The ABWC algorithm adjusts the backoff window to reflect the local network state in the local collision domain. Since the VMACs serve multiple rate-sensitive flows, it is possible

that more than one data frame collide with each other when their backoff times expire simultaneously. To solve such a virtual collision in the virtual collision domain, we designed the Virtual Collision Avoidance Control (VCAC).

The VCAC algorithm prevents the virtual collision by choosing a frame with the minimal cost when the virtual collision occurs in the virtual collision domain. By analyzing these algorithms, we proved that the ABWC component enables the achieved data rate to converge to the rate requirement and the VCAC component produces a virtual-collision-free schedule to avoid degradation of the achieved data rate.

The MSD model is easily integrated into the IEEE 802.15.4 MAC layer. The packet classifier relays the packets from the upper layer to the VMAC entities (See Figure 4.3). Basically, the packets are classified by the cross-layered packet classification. This classification method utilizes a pair of source and destination IP addresses in conjunction with a pair of source and destination ports, and puts them into the corresponding VMAC entities. The NSE component is responsible for estimating the network state and providing the achieved data rate. Since this component is coupled with the PHY layer, it utilizes the channel state information provided by the PHY layer. The VMAC entities implement the ABWC and VCAC algorithms. The ABWC component updates the achieved data rate and backoff window size every transmission, and the VCAC component works whenever the VMAC entities instantiate new packets for backoff.

Through the simulation, we validated our analysis and showed that the MSD outperforms existing algorithms.

5.2 Future Work

For the future work, we plan to investigate how beacon parameters, (BO,SO), has a critical impact on both the delay and the energy consumption. Basically, the length of the beacon interval and the superframe duration is determined by BO and SO, which are closely coupled with both the delay and the energy consumption. Especially, the SO determines the length of a GTS in the superframe. The short superframe duration may increase or decrease energy consumption depending on what amount of energy is consumed by wireless sensors.

To investigate the relationship between the delay and the energy consumption in sending delay-sensitive transactions, we will formulate an optimization problem of achieving the delay guarantee and minimizing the energy consumption. To formulate them, we will design an analytic model that describes the expected energy consumed by wireless sensors operating in reception, transmission and sleep states. We will provide an optimal solution method to solve the problem as well as extensive simulation results. This future work will give us the optimal beacon parameter, (BO,SO), that satisfies both the delay guarantee and minimizing the energy consumption simultaneously for delay-sensitive transactions.

For another future work, we plan to investigate three challenging problems. The first will be the problems of achieving the delay guarantee and maximizing the GTS utilization. The second will be addressed on the problems of minimizing the energy consumption and maximizing the GTS utilization. The third will be the problem of achieving the delay guarantee, minimizing the energy consumption and maximizing the GTS utilization.

For this purpose, we will not only design an analytic GTS utilization model that describes how much length of GTSs is exploited when the objective functions of the other problems are minimized, but also analyze it to identify what relationships between such two models exist. Through this analysis, a trade-off for solving the problems will be presented and discussed, and efficient solution methods will be developed and analyzed in terms of delay guarantee,

energy consumption and GTS utilization.

BIBLIOGRAPHY

[1] "IEEE Std 802.15.4:Wireless Medium Acess Control(MAC) and Physical Layer(PHY) Specifications for Low-Rate Wireless Personal Area Networks," LAN/MAN Standards Committee, 2006.

[2] "IEEE Std 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," LAN/MAN Standards Committee, 1999.

[3] E. Callaway, P. Gorday, and L. Hester, "Home Networking with IEE 802.15.4: A Developing Standard for Low-Rate Wireless Personal Area Networks," *IEEE Communications Magazine*, pp. 70–77, 2002.

[4] Y.-K. Huang, A.-C. Pang, and H.-N. Hung, "An Adaptive GTS Allocation Scheme for IEEE 802.15.4," *IEEE Trans. on Parallel and Distributed Systems*, vol. 19, pp. 641–651, 2008.

[5] G. C. Buttazzo, *Hard Real-Time Computing Systems*, 2nd ed. Springer, 2005.

[6] "http://www.wi-fi.org," Wi-Fi Alliance.

[7] "http://www.wirelesscommunication.nl/reference/chaptr01/wrlslans/hiperlan.htm."

[8] M. W. David J. Malan, Thaddeus Fulford-Jones and S. Moulton, "CodeBlue: An Ad Hoc Sensor Network Infrastructure for Emergency Medical Care," in *Proceeding of International Workshop on Wearable and Implantable Body Sensor Networks)*, April 2004.

[9] D. M. C. P. David C. Steere, Antonio Baptista and J. Walpole, "Research Challenges in Environmental Observation and Forecasting Systems," in *Proceeding of 6th International*

Conference on Mobile Computing and Networking (MobiCom2000)), August 2000, pp. 292–299.

[10] G. Lu, B. Krishnamachari, and et al., "Performance Evaluation of the IEEE 802.15.4 MAC for Low-rate Low-power Wireless Networks," in *IEEE International Conference on Performance, Computing, and Communications*, 2004, pp. 701–706.

[11] "IEEE Std 802.11e: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," LAN/MAN Standards Committee, 2005.

[12] J. F. Kurose and K. W. Ross, *Computer Networking:A Top-down Approach Featuring The Internet*, 3rd ed. Addison Wesley, 2005.

[13] A. Silberschatz and P. B. Galvin, *Operating System Concepts*, 4th ed. Addison-Wesley Publishing Company, 1994.

[14] L. F. Bic and A. C. Shaw, *Operating Systems Principles*. Prentice Hall, 2003.

[15] G. C. Buttazzo, "Rate Monotonic vs. EDF: Judgment Day," *Real-Time Systems*, vol. 29, pp. 5–26, 2005.

[16] A. Kanzaki and T. Uemukai, "Dynamic TDMA Slot Assignment in Ad hoc Networks," in *IEEE International Conference on Advanced Information Networking and Applications (AINA)*, March 2003, pp. 330–335.

[17] A. Koubâa, M. Aleves, and E. Tovar, "i-Game: An Implicit GTS Allocation Mechanism in IEEE 802.15.4 for Time-Sensitive Wireless Sensor Networks," 2006.

[18] A. Koubaa, M. Aleves, and E. Tovar, "GTS allocation analysis in IEEE 802.15.4 for real-time wireless sensor networks," in *IEEE International Parallel and Distributed Processing Symposium(IPDPS)*, April 2006, p. 158.

[19] L. Cheng, G. Bourgeois, and X. Zhang, "A New GTS Allocation Scheme for IEEE 802.15.4 Networks with Improved Bandwidth Utilization," in *IEEE International Symposium on Communications and Information Technologies*, Oct. 2007, pp. 1143 – 1148.

[20] T. Nadeem and A. Agrawala, "Efficient Time-Based Topology-Dependent Scheduling for Radio Packet Networks," in *IASTED International Conference on Communications and Computer Networks (CCN)*, vol. 1, November 2002.

[21] Y.-H. Tseng, E. H. kuang Wu, and et al., "Maximum traffic scheduling and capacity analysis for IEEE 802.15.3 high data rate MAC protocol," in *IEEE Vehicular Technology Conference*, vol. 3, October 2003, pp. 1678–1682.

[22] S. eun Yoo, D. Kim, and et al., "Scheduling support for guaranteed time services in IEEE 802.15.4 low rate WPAN," in *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, August 2005, pp. 400–406.

[23] T. H. Kim, D. H. Lee, and et al, "Priority Toning Strategy for Fast Emergency Notification in IEEE 802.15.4 LR-WPAN," in *Proceedings of Joint Conference on Communications and Information (JCCI)*, April 2005.

[24] A. Koubâa, M. Alves, M. Attia, and A. V. Nieuwenhuyse, "Collision-Free Beacon Scheduling Mechanisms for IEEE 802.15.4/Zigbee Cluster-Tree Wireless Sensor Networks," in *International Workshop on Applications and Services in Wireless Networks (ASWN)*, May 2007.

[25] S. B. Kodeswaran and A. Joshi, "Using location information for scheduling in 802.15.3 MAC," in *IEEE International Conference on Broadband Networks*, vol. 1, 2005, pp. 668–675.

[26] S. Cavalieri, S. Monforte, A. Corsaro, and G. Scapellato, "Multicycle Polling Scheduling Algorithms for FieldBus Networks," *Real-Time Systems*, vol. 25, no. 2-3, pp. 157–185, 2003.

[27] "Standard ECMA-368 High Rate Ultra Wideband PHY and MAC Standard," ECMA International, 2005.

[28] W. Ye, J. Heidemann, and D. Estrin, "Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493–506, 2004.

[29] "Network simulator ns-2," http://www.isi.edu/nsnam/ns/.

[30] J. Youn, S. Seok, and C. Kang, "New Service Differentiation Model for End-to-End QoS Provisioning in Wireless Ad Hoc Networks," *LNCS*, vol. 4104, pp. 376–389, July 2006.

[31] A. Veres, A. Campbell, M. Barry, and L. Sun, "Supporting Service Differentiation in Wireless Packet Networks Using Distributed Control," *IEEE Journal on Sel. Areas in Communications*, vol. 19, pp. 2081–2093, Oct. 2001.

[32] Y. Xiao, "A Simple and Effective Priority Scheme for IEEE 802.11," *IEEE Communications Letters*, vol. 7, no. 4, pp. 70– 72, Feb. 2003.

[33] ——, "Performance Analysis of Priority Schemes for IEEE 802.11 and IEEE 802.11e Wireless LANs," *IEEE Trans. on Wireless Communications*, vol. 4, no. 4, pp. 1506–1515, July 2005.

[34] Z. J. Haas and J. Deng, "On Optimizing the Backoff Interval for Random Access Schemes," *IEEE Trans. on Comm.*, vol. 51, pp. 2081 – 2090, Dec. 2003.

[35] F. Cali, M. Conti, and E. Gregori, "IEEE 802.11 Protocol: Design and Performance Evaluation of an Adaptive Backoff Mechanism," *IEEE Journal on Sel. Areas in Comm.*, vol. 18, pp. 1774–1786, Sep. 2000.

[36] ——, "Performance Modeling of an Enhanced IEEE 802.11 Protocol," in *Proc. of IFIP ATM*, June 1999.

[37] G.-S. Ahn, A. Campbell, A. Veres, and L.-H. Sun, "SWAN:Service Differentiation in Stateless Wireless Ad Hoc Networks," *IEEE Infocom 2002*, vol. 2, pp. 457 – 466, 2002.

[38] E. Kim, M. Kim, S. Youm, and S. Choi, "Multi-level Service Differentiation Scheme for the IEEE 802.15.4 Sensor Networks," *Lecture Notes in Computer Science (LNCS)*, vol. 3823, pp. 693 – 703, 2005.

[39] E. Kim, M. Kim, S. Youm, S. Choi, and C.-H. Kang, "Priority-based Service Differentiation Scheme for IEEE 802.15.4 Sensor Networks," *Elsevier's International Journal of Electronics and Communications*, vol. 61, pp. 69–81, 2006.

[40] A. Koubaa, M. Alves, B. Nefzi, and Y. Q. Song, "Improving the IEEE 802.15.4 Slotted CSMA/CA MAC for Time-Critical Events in Wireless Sensor Networks," in *International Workshop on Real Time Networks (RTN2006)*, Jul. 2006.

[41] D. Kipnis, A. Willig, J.-H. Hauer, and N. Karowski, "The ANGEL IEEE 802.15.4 Enhancement Layer: Coupling Priority Queueing and Service Differentiation," in *14th European Wireless Conference(EW2008)*, June 2008, pp. 1–7.

[42] E. A. Barbashin, *Introduction to the Theory of Stability*. Wolters-Noordhoff Publishing, 1970.

[43] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 3rd ed. John Wiley and Sons, 2006.

[44] D. C. Plummer, "An Ethernet Address Resolution Protocol," RFC826(tools.ietf.org/html/rfc826), November 1983.

[45] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," RFC3561(tools.ietf.org/html/rfc3561), July 2003.

[46] "IEEE Std 802.15.1: Wireless MAC and PHY Specifications for Wireless Personal Area Networks (WPANs)," LAN/MAN Standards Committee, 2002.

[47] "http://www.bluetooth.com," Bluetooth SIG.

[48] G. Bianchi, "Performance Analysis of the IEEE802.11 Distributed Coordination Function," *IEEE Journal of Selected Areas in Communications*, vol. 18, no. 3, pp. 535–548, March 2000.

[49] B. Li and R. Battiti, "Performance Analysis of an Enhanced IEEE 802.11 Distributed Coordination Functio Supporting Service Differentiation," in *International Workshop on Quality of Future Internet Service*, 2003.

[50] D. J. Inman, *Vibration with Control.* John Wiley and Sons, 2006.

[51] R. Shorey, A. Ananda, M. C. Chan, and W. T. Ooi, *Mobile, Wireless Sensor Networks:Technology, Applications, and Future Directions.* John Wiley and Sons, 2006.

[52] A. Halanay and V. Rasvan, *Applications of Lyapunov Methods in Stability.* Kluwer Academic Publishers, 1993.

[53] R. Jain, G. Babic, B. Nagendra, and C. Lam, "Fairness, call establishment latency and other performance metrics," *ATM Forum*, vol. TR 96-1173, Aug. 1996.

[54] L. A. Grieco, S. Mascolo, and R. Ferorelli, "Additive Increase Adaptive Decrease Congestion Control: A Mathematical Model and Its Experimental Validation," in *the 17th Intl. Symposium on Computers and Communications (ISCC'02)*, 2002.

[55] C. Liu and E. Modiano, "On the Performance of Additive Increase Multiplicative Decrease (AIMD) Protocols in Hybrid Space-terrestrial Networks," *Computer Networks*, vol. 47, pp. 661–678, April 2005.

[56] H. Morcos, I. Matta, and A. Bestavros, "M^2RC: Multiplicative-Increase/Additive-Decrease Multipath Routing Control for Wireless Sensor Networks," *SIGBED Rev.*, vol. 2, no. 1, pp. 13–18, Jan. 2005.

[57] S. H. Shah, K. Chen, and K. Nahrstedt, "Dynamic bandwidth management in single-hop ad hoc wireless networks," *Mob. Netw. Appl.*, vol. 10, no. 1-2, pp. 199–217, 2005.

[58] K. Xu, K. Tang, R. Bagrodia, M. Gerla, and M. Bereshinsky, "Adaptive Bandwidth Management and QoS Provisioning in Large Scale Ad hoc Networks," in *IEEE Military Communications Conference(MILCOM 2003)*, vol. 2, Oct. 2003, pp. 1018– 1023.

[59] C. Na, Y. Yang, and A. Mishra, "A Optimal GTS Scheduling Algorithm for Time-sensitive Transactions in IEEE 802.15.4 Networks," *Elsevier's Computer Networks*, vol. 52, no. 13, pp. 2543 – 2557, 2008.

[60] S. Singh and C. S. Raghavendra, "PAMAS: Power Aware Multi-Access Protocol with Signalling for Ad Hoc Networks," *ACM Computer Communication Review*, vol. 28, pp. 5–26, 1998.

[61] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides, "On the Construction of Energy-Efficient Broadcast and Multicast Trees in Wireless Networks," in 19^{th} *Annual Joint Conference of the IEEE Computer and Communications Societies(INFOCOM2000)*, vol. 2, March 2000, pp. 585 – 594.

[62] T. van Dam and L. Koen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *Proceedings of the International Conference on Embedded Networked Sensor Systems(SenSys)*. ACM, 2003, pp. 171 – 180.

[63] D. Li and X. Sun, *Nonlinear Integer Programming*. Springer, 2005.

[64] Z. Shelby, C. Pomalaza-raez, H. Karvonen, and J. Haapola, "Energy Optimization in Multihop Wireless Embedded and Sensor Networks," *International Journal of Wireless information Networks*, vol. 12, no. 1, pp. 11 – 20, 2005.

[65] CHIPCON, "cc2420: 2.4 GHz IEEE 802.15.4 / ZigBee-Ready RF Transceiver (Rev. B)," http://focus.ti.com/docs/prod/folders/print/cc2420.html.

[66] Atmel, "ATmega128L: 8-bit AVR Microcontroller with 128K Bytes In-System Programmable Flash," http://www.datasheetcatalog.com/datasheets-pdf/A/T/M/E/ATMEGA128L.shtml.

[67] A. Sinha and A. Chandrakasan, "Dynamic Power Management in Wireless Sensor Networks," *IEEE Design and Test of Computers*, vol. 18, no. 2, pp. 62 – 74, 2001.

[68] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks.* John Wiley & Sons, 2005.

[69] V. Rajendran, K. Obraczka, and J. G. luna aceves, "Energy-efficient, Collision-free Medium Access Control for Wireless Sensor Networks," *Wireless Networks*, vol. 12, pp. 63 – 78, 2006.

[70] Y. Sun, S. Du, O. GurewitzćÓ, and D. B. Johnson, "DW-MAC: A Low Latency, Energy Efficient Demand-Wakeup MAC Protocol for Wireless Sensor Networks," in *ACM International Symposium on Mobile Ad Hoc Networking and Computing(MobiHoc).* ACM, 2008, pp. 53–62.

[71] V. Rajendran, J. Garcia-Luna-Aveces, and K. Obraczka, "Energy-efficient, Application-aware Medium Access for Sensor Networks," in *IEEE Mobile Adhoc and Sensor Systems Conference(MASS)*, Nov. 2005, pp. 623–630.

[72] Q. Dong, "Maximizing System Lifetime in Wireless Sensor Networks," in *IEEE Information Processing in Sensor Networks (ISPN)*, April 2005, pp. 13 – 19.

[73] I. Kang and R. Poovendran, "Maximizing Network Lifetime of Broadcasting over Wireless Stationary Ad Hoc Networks," *Springer's Mobile Networks and Applications*, pp. 879 Ű– 896, 2005.

[74] A. Tiwari, P. Ballal, and F. L. Lewis, "Energy-Efficient Wireless Sensor Network Design and Implementation for Condition-Based Maintenance," *ACM Trans. on Sensor Networks (TOSN)*, vol. 3, no. 1, 2007.

[75] D. G. Luenberger, *Linear and Nonlinear Programming*, 2nd ed. Stanford University, 1989.

[76] I. Ramachandran, A. K. Das, and S. Roy, "Analysis of the Contention Access Period of IEEE 802.15.4 MAC," *ACM Transactions on Sensor Networks (TOSN)*, vol. 3, no. 1, 2007.

[77] H. A. Eiselt and C. L. Sandblom, *Integer Programming and Network Models.* Springer, 2000.

[78] S. Baase and A. V. Gelder, *Computer Algorithms: Introduction to Design and Analysis*, 3rd ed. Addison Wesley, 2000.

[79] C. Na and Y. Yang, "MRSD:Multirate-based Service Differentiation for the IEEE 802.15.4 Wireless Sensor Network," in *IEEE Global Communications Conference (Globecom)*, November 2009.

[80] C. Na, H. Cho, and et al., "Garbage Collector Scheduling in Dynamic, Multiprocessor Real-Time Systems," in *IEEE Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, August 2006, pp. 101–105.

[81] M. Neugebauer, J. Plonnings, and et al., "A New Beacon Order Adaptation Algorithm for IEEE 802.15.4 Networks," in *Proceedings of the European Workshop on Wireless Sensor Networks (EWSN)*, January 2005, pp. 302–311.

[82] C. Lu, J. Stankovic, and et al., "Design and Evaluation of a Feedback Control EDF Scheduling Algorithm," in *IEEE Real-Time Systems Symposium*, 1999.

[83] A. Torok, L. Vajda, and et al., "Techniques to improve scheduling performance in IEEE 802.15.3 based ad hoc networks," in *IEEE Global Telecommunications Conference (Globecom)*, vol. 6, November 2005.

[84] Z. Shao and U. Madhow, "A QoS Framework for Heavy-tailed Traffic over the Wireless Internet," in *IEEE Military Communications Conference (Milcom)*, vol. 2, October 2002, pp. 1201–1205.

[85] B. Manoj and C. S. R. Murthy, "Real-time Traffic Support for Ad hoc Wireless Networks," in *IEEE International Conference on Networks (ICON)*, 2002, pp. 335–340.

[86] G. Anastasi, M. Conti, M. D. Francesco, and A. Passarella, "Energy Conservation in Wireless Sensor Networks: A Survey," *Ad Hoc Networks*, vol. 7, pp. 537–568, 2008.

[87] P. Baronti, P. Pillai, V. Chook, S. Chessa, A. Gotta, and Y. F. Hu, "Wireless Sensor Networks: a Survey on the State of the Art and the 802.15.4 and ZigBee Standards," *Computer Communications*, vol. 30, no. 7, pp. 1655–1695, 2006.

[88] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *Proceedings of the International Conference on Embedded Networked Sensor Systems(SenSys)*. ACM, November 2004, pp. 95–107.

[89] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.

[90] S. Doshi, S. Bhandare, and T. X. Brown, "An On-demand Minimum Energy Routing Protocol for a Wireless Ad Hoc Network," *SIGMOBILE Mobile Computing and Communications Review*, vol. 6, no. 3, pp. 50–66, 2002.

[91] S. Doshi, "Multi-constrained Quality of Service Aware Routing in Mobile Ad Hoc Wireless Networks," Ph.D. dissertation, Boulder, CO, USA, 2005, director-Brown, Timothy X.

[92] Q. Li, J. Aslam, and D. Rus, "Online Power-aware Routing in Wireless Ad-hoc Networks," in *International Conference on Mobile Computing and Networking (MobiCom)*. ACM, 2001, pp. 97–107.

[93] J.-H. Chang and L. Tassiulas, "Maximum Lifetime Routing In Wireless Sensor Networks," *IEEE/ACM Transactions on Networking*, vol. 12, pp. 609–619, 2000.

[94] S. Ergen, C. Fischione, D. Marandin, and A. Sangiovanni-Vincentelli, "Duty-Cycle Optimization in Unslotted 802.15.4 Wireless Sensor Networks," in *IEEE Global Telecommunications Conference (Globecom)*. IEEE, 2008, pp. 1–6.