A Novel Scheduling Algorithm for Video Flows in High-rate WPANs

by

Shahab Moradi

B.Sc., Electrical Engineering, University of Tehran, Iran, 2004

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Master of Applied Science

in

The Faculty of Graduate Studies

(in Electrical and Computer Engineering)

The University of British Columbia

April 2007

Abstract

The emerging high-rate wireless personal area network (WPAN) technology is capable of supporting high-speed and high-quality real-time multimedia applications. In particular, video streams are deemed to be a widespread traffic type, and require quality of service (QoS) support. However, in the current IEEE 802.15.3 standard for MAC (media access control) of high-rate WPANs, the implementation details of some key issues such as scheduling and QoS provisioning have not been addressed. Moreover, the hierarchical structure of video streams calls for special measures at the MAC layer in order to improve the QoS.

In the first part of this thesis, we propose a frame-decodability aware (FDA) technique to make the scheduling algorithms aware of the hierarchical structure and decoding dependencies in video streams. Simulation results show that the FDA technique can significantly improve the performance of F-SRPT [1] and EDD+SRPT [2] schedulers by up to 61% and 60%, respectively. We also compare two common performance metrics and investigate which one is a more accurate indicator of the QoS given to video streams.

In the second part of this thesis, we first propose a mathematical model for the optimal scheduling scheme for video flows in high-rate WPANs. Using this model, we then propose

a scheduler that incorporates reinforcement learning (RL). Simulation results show that our proposed scheduler is nearly optimal and performs 42%, 49%, and 53% better than EDD+SRPT [2], PAP [3], and F-SRPT [1] schedulers, respectively.

Table of Contents

List of Tables

List of Figures

List of Acronyms

ACK	Acknowledgement
ATP	Association Timeout Period
BcstID	Broadcast Identifier
BER	Bit Error Rate
CAP	Contention Access Period
CCA	Clear Channel Assessment
CMAC	Cerebellar Model Articulation Controller
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CTA	Channel Time Allocation
CTAP	Channel Time Allocation Period
DC	Direct Current
DCM	Darken Chang Moody

DEV	Device
DEVID	Device ID
DFR	Decoding Failure Rate
Dly-ACK	Delayed Acknowledgement
DP	Dynamic Programming
DstID	Destination Identification
DS-UWB	Direct Sequence Ultra-wide Band
ECMA	European Computer Manufacturers Association
EDD	Earliest Due Date
FCTA	Feedback Channel Time Allocation
FDA	Frame-decodability Aware
F-SRPT	Fair Shortest Remaining Processing Time
GOP	Group of Pictures
ID	Identification
IE	Information Element
IEEE	Institute of Electrical and Electronics Engineers

IFFT	Inverse Fast Fourier Transform
Imm-ACK	Immediate Acknowledgement
JFR	Job Failure Rate
MAC	Media Access Control
MB-OFDM	Multi-band Orthogonal Frequency Division Multiplexing
MCTA	Management Channel Time Allocation
MDP	Markov Decision Process
MIFS	Minimum Inter-frame Spacing
MUI	Multiuser Interference
PAP	Pre-assigned Priority
PNC	Piconet Coordinator
QoS	Quality of Service
USB	Universal Serial Bus
UWB	Ultra-wide Band
RF	Radio Frequency
RL	Reinforcement Learning

SIFS	Short Inter-frame Spacing
SINR	Signal to Interference and Noise Ratio
SMART	Semi-Markov Average Reward Technique
SMDP	Semi-Markov Decision Process
SrcID	Source Identification
SRPT	Shortest Remaining Processing Time
TDMA	Time Division Multiple Access
TG	Task Group
TFC	Time-frequency Coding
TH	Time Hopping
TH-UWB	Time Hopping Ultra-wide Band
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network

List of Symbols

$\lvert \cdot \rvert$	cardinality of a set
$\lfloor \cdot \rfloor$	floor function
$\setminus$	set difference
$\leftarrow$	assignment
$\triangleq$	definition
$\boldsymbol{A}$	set of all actions
$\boldsymbol{A}_{\mathbf{s}}$	set of all possible actions in state $\mathbf{s}$
a_i	scheduling decision about flow i
$\mathbf{a}$	a possible action
β	similarity radius
B	B frame type of video flows
$c_i(\mathbf{s})$	state-dependent cost function for flow i

CR	cumulative reward
CT	cumulative time
δ_i	decodability of the frame of flow i
$\boldsymbol{\delta}$	decodability vector
d_i	deadline of the frame of flow i
d_i^{new}	deadline of the newly arrived frame for flow i
$\mathbf{d}$	vector of deadlines
$\text{dist}(\mathbf{x}, \mathbf{z})$	distance of feature vectors $\mathbf{x}$ and $\mathbf{z}$
D^{max}	maximum deadline
$\overline{\text{DFR}}^{\pi}$	average DFR that policy π obtains
η	superframe size
e_i	eligibility of the frame of flow i
$f(\mathbf{s}, \mathbf{a})$	feature vector of state $\mathbf{s}$ and action $\mathbf{a}$
F	number of flows in the system
$\mathcal{F}$	set of integers from 1 to F
γ	inter-arrival time of video frames

g_i	offset of the frame of flow i with respect to the beginning of the frame's GOP
$\mathbf{g}$	vector of g_i's
$\mathbf{h}_k$	kth prototype of Kanerva coding
$H_{\mathbf{x}}$	set of prototypes activated by feature vector $\mathbf{x}$
$\lambda(\mathbf{x}, \mathbf{z})$	similarity between feature vectors $\mathbf{x}$ and $\mathbf{z}$
l_i	channel time that flow i requires
l_i^{new}	channel time required by the newly arrived frame for flow i
$l^{partial}$	amount of channel time allocated for partial transmission of a frame
$\mathbf{l}$	vector of channel time requirements
L_i^{max}	maximum frame size of flow i
I	I frame type of video flows
K	minimum number of prototypes that should be activated by a feature vector
μ_0, μ_r	parameters of DCM for learning rate
μ_n	learning rate at superframe n

M	I-to-P distance of a video flow
M_i	I-to-P distance of flow i
n	superframe number
(N, M)	GOP pattern
N	I-to-I distance of a video flow
N_i	I-to-I distance of flow i
$O(\cdot)$	big O function
ϕ	start time separation
p	index of the flow that partially transmit its frame
P	P frame type of video flows
q_0, q_r	parameters of DCM for exploration probability
q_n	exploration probability at superframe n
$\hat{Q}(\mathbf{x})$	value of feature vector $\mathbf{x}$ estimated by Kanerva coding
ρ^π	gain of policy π
$r(\mathbf{s}, \mathbf{a})$	reward of taking action $\mathbf{a}$ in state $\mathbf{s}$
$\mathbb{R}$	set of real numbers

$\mathbf{s}$	a possible state
$\boldsymbol{S}$	set of all the states
$\mathrm{tx}_i(x)$	channel time that flow i requires for transmitting x bits of data
T	last decision epoch
$U_{\{\cdot\}}$	indicator function
w_k	weight of the kth prototype
y_i	type of the frame of flow i
$y_i(g)$	type of the frame of flow i that has offset g
y_i^{new}	type of the newly arrived frame for flow i

Acknowledgements

It is difficult to overstate my gratitude to my Master's supervisor Dr. Vincent Wong for his guidance, patience, and support during my graduate studies. I sincerely appreciate the amount of time and effort he invested in helping me developing the ideas, improving the results, and refining the structure of my thesis.

I am indebted to Dr. Richard Sutton for his insightful comments, Dr. Pentti Kanerva for providing me with his publications, and Hamed Mohsenian for his helpful suggestions. I would also like to thank my fellow colleagues and friends who assisted and encouraged me along the way. I am especially grateful to Adrienne Peltonen, Amir Moghimi, Farshid Farhat, Babak Fallah, Samaneh Ashoori, Enrique Stevens-Navarro, Stephen Ney, Jessica Raposo, Mohammad Maysami, Ali Soltanzadeh, Hamidreza Ahmadian, Ashkan Heshmati, Bijan Azadi, Mehran Shaghaghi, and Nasim Arianpoo.

Lastly, and most importantly, I wish to thank my parents, Mohammad-Taghi and Marzieh Moradi, as well as my sister and her spouse, Sholeh and Mohammad Adimi, for their love, support, and encouragement. To them I dedicate this thesis.

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant number CRDPJ 320552-04.

Chapter 1

Introduction

In the past few years, ultra-wide band (UWB) technology has received increasing attention in the wireless world. It provides short-range connectivity, low transmit power levels, and high-data rates, which make UWB the physical layer of choice for high-rate wireless personal area networks (WPANs). UWB-enabled WPANs can offer many new applications, such as home entertainment, real-time multimedia streaming, and wireless USB, to name a few.

In order to fully exploit UWB technology in high-rate WPANs, upper layers, including the media access control (MAC) layer, must be properly designed for high-rate applications. Video transmission is one such application for high-rate WPANs, which is predicted to constitute a major traffic load. Real-time video flows are delay-sensitive and require quality of service (QoS) guarantee. However, in the IEEE 802.15.3 standard for MAC [5], which is designed for WPANs, details of scheduling and QoS support are left to the developers. Consequently, in this thesis, we aim to design an application-aware scheduling algorithm for MAC layer to provide the required QoS for video traffic.

1.1 Motivations and Objectives

Similar to other real-time traffic, video is delay sensitive and its frames are dropped at the receiver if their delay exceeds the maximum tolerable delay. However, video stream has a few unique characteristics that make QoS support more challenging than other real-time traffic. It has large peak-to-average ratio of the frame sizes and hierarchical structure with dependency among its frames [3].

The related work on the video traffic schedulers in high-rate WPANs is limited. The few existing proposals try to enhance the scheduling performance by using some kind of application layer information about video streams. This information is mainly the type of the video frames. However, the information about frame type is not used efficiently because it can be *readily* used along with the decoding dependencies in video streams to further improve scheduling performance. We aim to design a technique that enables the scheduler to be aware of the decoding dependencies, and thus, use the resources more efficiently. Furthermore, this technique must 1) be simple to implement, 2) not incur signaling overhead in the system, and 3) be independent of the scheduling algorithm details.

Besides not fully using application layer information, all the current proposals are heuristics that describe how application layer information can be combined with a conventional scheduling algorithm for better performance. As a result, we are motivated to devise an analytical framework to provide us with a foundation to design a systematic scheduling algorithm for video traffic.

1.2 Contributions

Our contributions in this thesis can be summarized as follows:

- We propose a frame decodability aware (FDA) technique that the scheduler can use to determine whether a video frame is decodable at the receiver. The FDA technique uses the information about frame type to minimize the channel time wastage caused by scheduling undecodable frames. This technique is applicable to different schedulers and can improve the QoS [6].

- We provide a mathematical framework for finding the optimal scheduler of video flows in high-rate WPANs. This framework takes into account the number and pattern of video flows, and their hierarchical structure.

- Based on the above framework, we design a scheduler that provides significantly better QoS to video flows when compared to some other schedulers [7].

1.3 Structure of the Thesis

In Chapter 2, we give an overview of UWB physical layer and IEEE 802.15.3 standard for MAC. It also summarizes the hierarchical structure of video streams and the related work. In addition, Chapter 2 introduces Markov decision process (MDP) and reinforcement learning (RL), which lays the foundation for Chapter 4. In Chapter 3, we describe our proposed FDA technique which can efficiently use application layer information to

improve performance. The mathematical formulation for optimal video scheduler, as well as our proposed scheduling algorithm for video traffic, are presented in Chapter 4. The conclusions in Chapter 5 outline the main contributions of the thesis and explain possible future works in the area.

Chapter 2

Background and Related Work

In recent years, many researchers have addressed the scheduling and QoS support issues in high-rate WPANs, which are not specified in the standard for MAC [5]. The different types of physical layer for UWB transmission, as well as attention to different details by which performance improvements can be achieved, have resulted in a rich literature in this area.

In this chapter, we first give an overview of the types of UWB physical layer, followed by a brief description of the IEEE 802.15.3 standard for MAC. Since real-time traffic is envisioned to constitute a major traffic load in WPANs, challenges to provide QoS to this traffic type are also addressed in this chapter. After that, we summarize the key proposals for MAC scheduling algorithms for WPANs. Since we use Markov decision process (MDP) and reinforcement learning (RL) in Chapter 4, we introduce the basics of MDP and RL in this chapter as well.

2.1 Physical Layer

UWB signal is defined as a signal having a -10 dB fractional bandwidth greater than 20%, or occupying at least 500 MHz of bandwidth. This section introduces the main

types of UWB physical layer. The first type is impulse-based UWB which can further be classified as time-hopping (TH)-UWB and direct sequence (DS)-UWB. The former is used for low-rate WPANs, while the latter is suitable for high-rate WPANs. The second type of UWB physical layer, which is a contender of DS-UWB for physical layer of high-rate WPANs, is multi-band orthogonal frequency-division multiplexing (MB-OFDM). We briefly describe these types in the following subsections.

2.1.1 Impulse-based UWB

Impulse-based UWB systems utilize pulses with duration ranging from a few tens of picoseconds up to a few nanoseconds, which have frequency components from near direct current (DC) to a few giga Hertz [8]. The typical waveforms used in impulse-based UWB are Gaussian monocycle (the first derivative of Gaussian pulse with the form $\alpha\, t \exp(-t^2/\tau^2)$, where t is the time, τ determines the pulse duration and α is the amplitude coefficient) and its derivatives. Standard modulation schemes such as pulse amplitude, position or phase modulation can be used.

The two types of impulse-based UWB systems are TH-UWB and DS-UWB. Using the notations in [8], a user of DS-UWB sends a bit in an interval with duration T_f. Within this period, each user uses a pseudo random spreading code with length N_s and chip duration T_c to spread its data. Thus, the data rate for each user is $1/T_f$. On the other hand, a user of TH-UWB spreads one bit to N_s intervals each with duration T_f. The pseudo random time-hopping code assigned to the user determines the pulse position

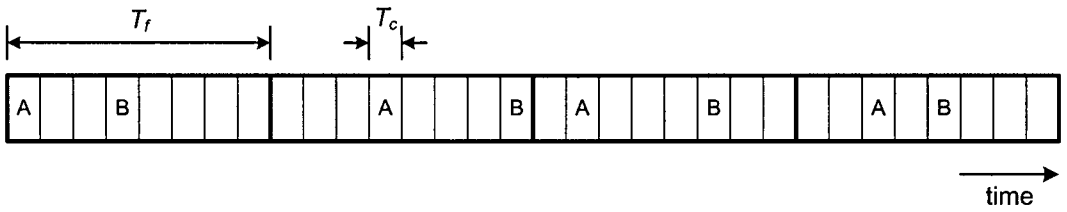

Figure 2.1: Multiple access using time-hopping with $N_s = 4$ and $N_h = T_f/T_c = 8$. Users A and B use 0, 3, 1, 2 and 3, 7, 5, 4 TH codes, respectively.

within each interval. Therefore, the data rate for each user is $1/N_sT_f$. The use of pseudo random codes can be used for multiple access. Figure 2.1 illustrates this for TH-UWB.

Impulse-based UWB system has two main advantages. First, it is robust to multipath propagation as most of the multipaths are resolvable. Second, pulses with short duration can give accurate timing information that can be used for ranging and positioning [9]. However, very short duration of pulses requires very long acquisition time; thus, reducing synchronization overhead becomes a challenge. Besides, suppressing narrowband interference requires notch filters, which cause additional complexity to compensate the distortion of the pulses caused by the notch filters [10].

2.1.2 Multiband OFDM (MB-OFDM)

As its name suggests, MB-OFDM divides the available spectrum into 14 sub-bands with 528 MHz bandwidth, and uses OFDM modulation within each sub-band. The spectrum is divided into five band groups. Each of the first four band groups has three sub-bands, whereas the fifth one has only two sub-bands (see Figure 2.2). The MB-OFDM system utilizes a time-frequency code (TFC) to interleave coded data over sub-bands of each

band group. As shown in Table 2.1, there are four TFCs available to the 3-band groups, and two TFCs available to the only 2-band group. Together, these band groups and the TFCs define eighteen separate logical channels that can be used by independent piconets. TFCs are designed to have good collision properties for all possible asynchronous shifts among the simultaneously operating piconets. All devices should be able to operate in the band group #1, whereas support for the other band groups is optional and can be added in the future [4].

Since UWB regulations may vary worldwide, the MB-OFDM approach gives a high regulatory flexibility for worldwide operation by simply turning different bands on or off. Furthermore, MB-OFDM is flexible in coexisting with other systems in uncoordinated environments, by selectively using available bands according to the in-band interference level. For instance, UWB systems are likely to receive interference from IEEE 802.11a operating at 5 GHz. This interference can be reduced by not using the sub-bands overlapping with the operating spectrum of IEEE 802.11a. In addition, it can efficiently capture multipath energy in a single radio frequency (RF) chain [11]. However, the MB-OFDM transmitter is slightly complex because it uses an inverse fast Fourier transform (IFFT) [12].

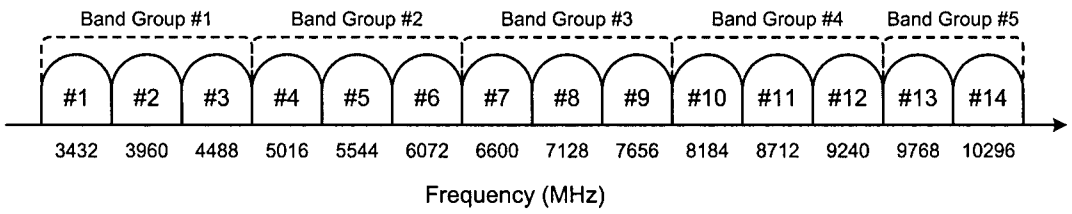

Figure 2.2: Diagram of the sub-bands and band groups of MB-OFDM [4].

TFC Number	Length 6 TFC					
1	1	2	3	1	2	3
2	1	3	2	1	3	2
3	1	1	2	2	3	3
4	1	1	3	3	2	2
5	1	2	1	2	1	2
6	1	1	1	2	2	2

Table 2.1: Time Frequency Codes (TFC) used in MB-OFDM. The first four TFCs are used for the 3-band groups, and the last two TFCs are for the 2-band group.

2.2 IEEE 802.15.3 Standard for MAC

Personal operating space is a space around a person or an object with a typical radius of up to 10 m [5]. WPANs are used to convey information within personal operating space. As a result, unlike wireless local area networks (WLANs), link robustness at long range is not a primary concern in the development of WPAN standards. This allows the standard to primarily focus on issues such as cost, size, power consumption and data rate. The IEEE 802.15.3 Task Group has designed the MAC standard for WPANs, which aims to provide low cost, low power consumption, and high data rate within the personal operating space. In the following subsections, we describe the main features of the standard based on [5].

2.2.1 Communication Structure

According to [5], a piconet is defined as a "wireless ad hoc data communications system which allows a number of independent data devices (DEVs) to communicate with each other." The size of a piconet is generally confined to the personal operating space which is much smaller than the coverage area of local or metropolitan area networks, and hence the name. Figure 2.3 shows the different elements (DEVs and PNC) of an 802.15.3 piconet, hereafter briefly referred to as a piconet. As illustrated, communication within a piconet is based on central coordination and peer-to-peer data transfer. One DEV called the piconet coordinator (PNC) is responsible for providing basic timing, performing scheduling, managing QoS, and controling access to the piconet. MAC overhead, including headers and acknowledgement frames, is normally transmitted at a lower rate, called the base rate, to guarantee reliable detection at the receiver.

2.2.2 Piconet Initiation, Maintenance and Termination

Since a piconet is a kind of ad hoc network, and thus infrastructure-less, a procedure is required to create a piconet. In this procedure, a DEV, which is capable of performing the roles of PNC, scans the available channel. The number of available channels depends on the underlying physical layer and is specified in the standard. If the PNC finds an unused channel, it starts sending beacons in that channel. Otherwise, it can associate with an existing piconet. This starting procedure does not guarantee that the most capable DEV will be the PNC. Therefore, a PNC handover procedure is also defined which is described

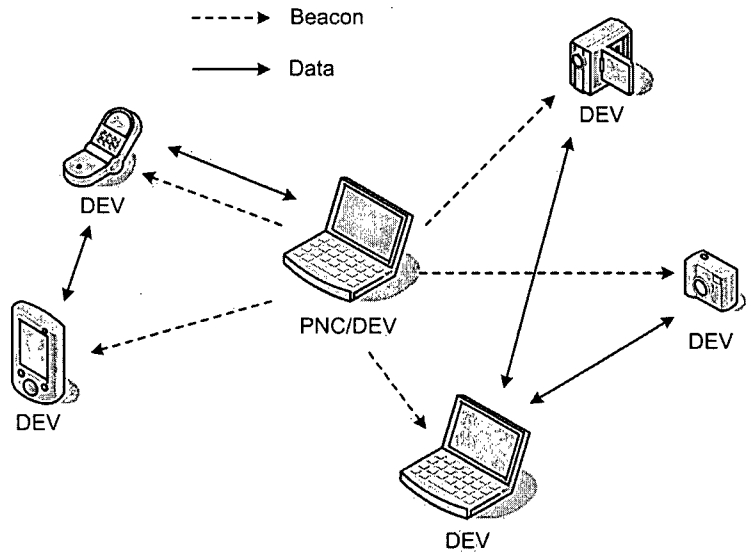

Figure 2.3: Elements of IEEE 802.15.3 piconet

later in this section. A DEV can join or leave a piconet by using associate or disassociate procedures, respectively. After association, DEV is provided with a unique one-octet ID, called DEVID. This DEVID is used instead of DEV's 8-octet address in order to reduce the overhead. This imposes a limit on the maximum number of DEVs in a piconet. When a new DEV associates with the piconet, the PNC checks its capabilities. If it is more capable than the PNC itself and the current security policy also allows handover, the PNC can handover the control of the piconet to that DEV. Several criteria, such as being able to support security, are considered to determine the capabilities. When a DEV disassociates from the piconet, its DEVID is no longer valid. The PNC does not re-assign it until expiration of a waiting period. When the PNC wants to shutdown, it handovers the control to another PNC-capable DEV in the piconet; otherwise, it sends a shutdown

information element (IE) to inform the members of the piconet. If the PNC abruptly leaves, the piconet terminates its operation for the duration of association timeout period (ATP). After expiration of ATP, another PNC-capable DEV can start a piconet using the procedure described earlier in this section.

2.2.3 The Superframe Structure

The timing in the piconet is based on the superframe, which consists of three parts (see Figure 2.4) as follows:

1. Beacon – It announces timing allocations and is also used to communicate management information for the piconet. It contains piconet synchronization parameters, such as superframe duration and contention access period end time. It may also include several IEs, such as channel time allocation, DEV association, PNC shutdown, or PNC handover.

2. Contention Access Period (CAP) – The presence of this part is optional. It is used to communicate commands and asynchronous data. It is stated in the beacon the types of command and data that are allowed to be transmitted during CAP. Some examples of command are association request/response, PNC handover request/response, and channel time request/response. The access method within CAP is carrier sense multiple access with collision avoidance (CSMA/CA) with binary exponential backoff. The length of CAP is determined by the PNC and may vary from one superframe to another.

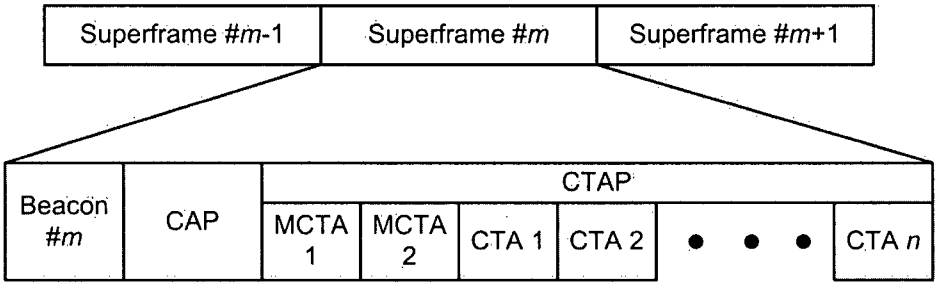

Figure 2.4: Superframe structure. The presence of CAP is optional. The number, duration and order of the CTAs and MCTAs can vary from one superframe to another.

3. Channel Time Allocation Period (CTAP) – This period is composed of channel time allocations (CTAs), and may include management CTAs (MCTAs). When a DEV is assigned a CTA, it can use it for commands, isochronous streams, and asynchronous data transfer. Channel access during CTAP is time division multiple access (TDMA) and there is no contention within this time. At the beginning of the superframe, the PNC announces the start time and length of each CTA as well as the DEVs that are allowed to use it (typically, source and destination of a flow, which are referred to as SrcID and DestID. In the case of broadcast or multicast, more DEVs are involved.) This will allow DEVs to operate in normal power consumption mode only during their CTA and switch to low power consumption mode throughout the rest of CTAP. This scheme also allows the PNC to support QoS. If either the source or destination of a CTA is the PNC, that CTA is called MCTA. The PNC has the option of using MCTAs instead of CAP for sending or receiving command frames. Like other CTAs, the number and position of MCTAs may vary from superframe to superframe and the appropriate number is determined by the

PNC. MCTAs can be assigned to a specific source or destination, and function as direct uplink or downlink, respectively. For example, in the superframe following the successful association of a new DEV, the PNC can allocate an MCTA to that DEV. Thus, it is guaranteed that the newly associated DEV can communicate its traffic needs with the PNC as soon as possible, which in turn, results in fast connection to the piconet. It is also possible that an MCTA be shared among several DEVs, as in *open* and *associate* MCTAs. An MCTA is open if its SrcID is the broadcast identifier (BcstID). Therefore, all associated DEVs can send command frames to the PNC during open MCTA. On the other hand, the SrcID of associate MCTA is UnassocID, which means that all DEVs that are not currently associated in the piconet can attempt to send association request to the PNC in order to become a member of the piconet. Channel access during shared MCTA is slotted ALOHA.

2.2.4 Data Communication Between DEVs

As mentioned in Section 2.2.1, all data exchange is peer-to-peer. Furthermore, CAP can be used to send asynchronous data, and CTAP is used for both isochronous streams and asynchronous data transfer. Isochronous streams require channel time on a regular basis, whereas in the case of asynchronous data transfer, only the total amount of required channel time is important. The PNC can split this amount among several superframes in any manner, as long as they sum up to the requested amount. According to its current

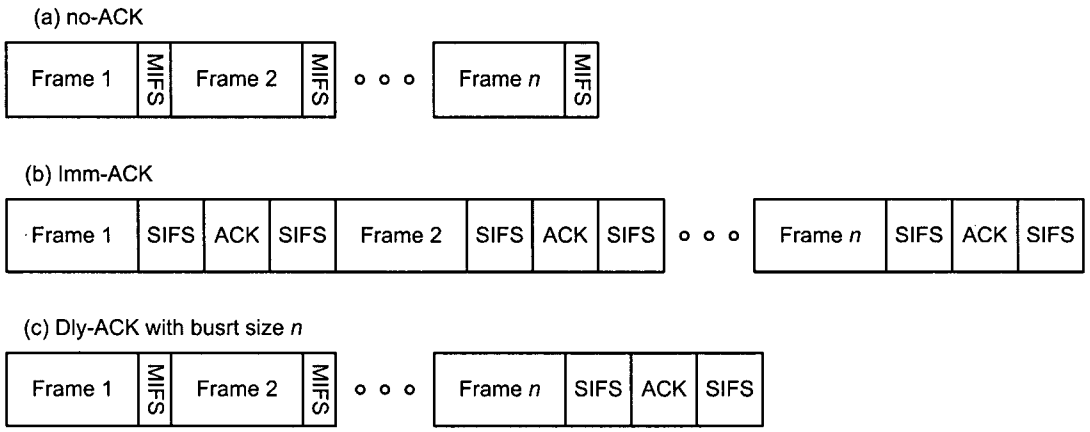

Figure 2.5: ACK policies defined in IEEE 802.15.3 Standard for MAC: (a) no-ACK, (b) Imm-ACK, (c) Dly-ACK.

needs, each DEV can send a CTA request to the PNC to modify the length or number of its allocated CTAs. However, the PNC may or may not be able to fulfill all the requests of the DEVs. In any case, the PNC will notify the DEV of its decision, through the channel time response command.

Three acknowledgement (ACK) policies are defined for MAC (see Figure 2.5): no-ACK, immediate ACK (Imm-ACK) and delayed ACK (Dly-ACK). The No-ACK policy is used for applications that do not require guaranteed delivery, and two successive frames are separated by minimum interframe space (MIFS). On the other hand, when Imm-ACK is used, every frame should be acknowledged upon reception. The short interframe space (SIFS) separates the transmitted frame and ACK. Dly-ACK is a hybrid of the two previous policies. In this mechanism, the receiver groups the ACKs of a burst of frames and sends them all to the source in one ACK frame. Adjusting the burst size can balance between overhead and transmission reliability.

2.3 QoS Issues

Providing guaranteed QoS is a crucial issue in the design of networks that are aimed to serve different types of traffic including multimedia and real-time traffic. For the QoS traffic classes, networks should be able to, deterministically or statistically, satisfy some performance criteria, such as minimum bandwidth allocation or bounded transmission delay. QoS management methods include resource reservation, admission control, QoS routing, and packet scheduling, many of which are related to MAC protocol [13]; hence, MAC design is of vital importance to provide QoS.

Multimedia traffic, and more specifically video traffic, requires QoS support in high-rate WPANs. Besides typical challenges to meet QoS requirements of multimedia traffic, such as strict delay and loss bounds, video stream introduces other complexities. It has large peak-to-average ratio of the frame sizes and hierarchical structure with dependency among its frames [3]. Consequently, it is essential to study the performance of proposed scheduling algorithms for IEEE 802.15.3 under video flows.

In the following subsections, we first describe different video frame types and hierarchical structure of video streams. We then introduce the performance metrics that will be used in Chapters 3 and 4.

2.3.1 Characteristics of Video Streams

A typical video encoder generates a sequence of three types of compact frames: intra-coded (I), predictive (P), and bidirectional (B) frames [14]. Because of the different

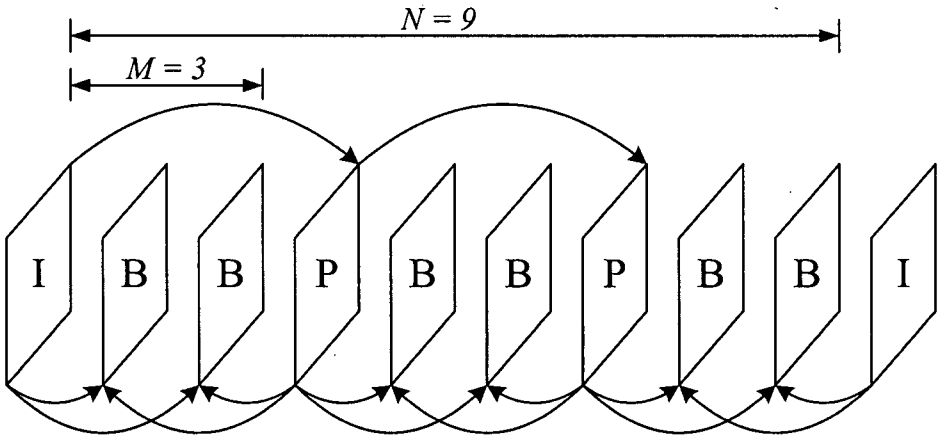

Figure 2.6: GOP structure for (N=9,M=3). The arrows show the decoding dependencies.

compression schemes used to encode the different frame types, I frames *tend* to be larger (less compressed) than P and B frames, and P frames *tend* to be larger than B frames. Video encoders generate the three frame types according to a predefined pattern called group of pictures (GOP). This pattern is characterized by two parameters (N, M), where N is the I-to-I frame distance, and M is the I-to-P frame distance [14]. This pattern is generally fixed for a given video sequence, and N is a multiple of M. Figure 2.6 illustrates the hierarchical structure of video streams, as well as decoding dependencies among the frames. In order for a frame to be decodable at the receiver, all other frames that it depends on, including the frame itself, must be available at the receiver. A frame is undecodable if it is *directly* or *indirectly lost* [15]. Direct loss happens when the receiver does not receive the frame completely, either because of channel error or deadline expiration. A frame is indirectly lost when some frames that it depends on are directly

Frame type	Number of frames depending on it
I	$N + (M - 1)$
$P_k, k = 1, \ldots, \frac{N}{M} - 1$	$N - 1 - (k - 1)M$
B	1

Table 2.2: Number of frame dependencies for each video frame

lost. As an example, in Figure 2.6, all the frames in a GOP depend on the only I frame in that GOP; therefore, if that I frame does not meet its deadline, the whole GOP is considered undecodable (direct loss of the I frame and hence, indirect loss of the rest of the frames). For instance, with $N = 12$ and frame rate of 30 frames per second (fps), this would result in impaired quality for about 360 ms, which is fairly obvious to a user. Similarly, loss of P frames can also degrade the quality of video streams. Table 2.2 shows the number of frames depending on each frame inside a GOP. The larger the number of dependencies on a frame, the more important that frame is for decoding process at the receiver. Consequently, video frame types can be ranked, from the most important to the least important, as I, P, and B.

2.3.2 Performance Metrics

Video frames, like other real-time traffic, are delay sensitive. The video frames will be dropped at the receiver if their delay exceeds the maximum tolerable delay. This is the base of *job failure rate* (JFR) criterion for evaluating performance of schedulers in the MAC layer. For a delay sensitive flow, JFR is defined as the fraction of frames that

do not meet their transmission deadlines, and are hence useless and get dropped. An important consequence of hierarchical structure of video stream is that JFR, by itself, cannot accurately reflect the QoS given to a video flow at the MAC layer. In other words, low JFR does not necessarily indicate high QoS. As a result, another performance metric is required for comparing the performance of different schedulers. We use the *decoding failure rate* (DFR) criterion, which takes frame dependencies into account. It is defined as the ratio of the total number of undecodable frames to the total number of frames [15]. DFR can be viewed as an objective measure of user-perceived degradation of quality, and is the main performance metric in this thesis.

2.4 Related Work on MAC Enhancement

This section gives an overview of the related work on scheduling algorithms for MAC in WPANs. In Section 2.4.1, we introduce the proposals that incorporate the unique properties of impulse-based UWB physical layer, especially the possibility of simultaneous transmissions. In Section 2.4.2, we summarize those proposals that are applicable to various types of UWB physical layer.

2.4.1 Scheduling Proposals for Impulse-based UWB

Both DS-UWB and TH-UWB have the potential to support concurrent transmissions, by using time hopping and spreading codes, respectively. Many researchers have tried to take advantage of this property and improve the efficiency of resource allocation. The

scheduling scheme proposed in [16] chooses the appropriate scheduling method out of single-link transmission policy and concurrent-links transmission policy, based on the power level of the active links. Since transmission power in UWB networks is very low, two links with a large separation in space will cause negligible interference to each other and thus can be active at the same time. To clarify such large space separation, a concept, called exclusive region, is defined in [17] and [18], which is used to choose between concurrent and single transmission policies. Another approach is the resource allocation algorithm proposed in [19]. It uses K-coloring problem from graph theory to perform scheduling. This problem is that, given a graph G with V vertices and E edges, find the minimum number of colors for coloring the vertices in such a way that no adjacent vertices (i.e., connected with an edge) have the same color. Links, time slots, and the number of time slots in a superframe can be mapped to the vertices, colors, and the number of colors in the K-coloring problem. The edges also represent heavy interference between two links. The output of this algorithm is the list of links that are allowed to transmit in each of the time slots. If more than one link is scheduled in a time slot, they will transmit concurrently. Shen *et al.* used a different approach in [20]. They define a weight for each link that is a measure of potential interference on that link. In other words, the larger the weight is, the more the links that can interfere with that link will be. These weights are then used by an algorithm called maximal weighted matching to group flows into concurrent transmission groups. They also consider the fact that not all the flows that are scheduled to transmit concurrently during a time slot use the whole

time slot. By taking the duration of each transmission into account, they reuse these idle times to further enhance the throughput and channel reuse.

Concurrent transmission causes multi-user interference (MUI), which can limit the capacity of the system. Similar to other spread spectrum systems such as CDMA, power allocation can be used in impulse-based UWB systems to combat MUI. However, in UWB systems, under some conditions, power control is not an efficient way to guarantee the required signal to interference and noise ratio (SINR) at the receiver [11]. An effective way to compensate for the shortcomings of power control is rate control. There are two main ways to adjust the rate of a node in an impulse-based UWB system. The first method is to use adaptive channel coding [17], where the code rate is adapted to the channel condition. Based on the feedback information that the sender obtains from the receiver, the sender uses a lower (higher) rate channel code if the channel becomes worse (better). An efficient way of implementing this adaptive channel coding is to use codes that provide incremental redundancy. In these codes, a high-rate code is a subset of lower rate codes. Therefore, additional protection can be provided to high-rate codes by only transmitting the extra redundancy and not the whole data encoded with the stronger code. The second method is to alter TH sequence parameters, i.e. number of pulses for each bit (N_s), maximum time hopping shift (N_h), and time hopping unit (T_c) [19]. This is a unique control method inherent in impulse-based UWB networks.

Combined with rate allocation, the power allocation problem can be formulated as a joint optimization problem so as to minimize the total power consumption [19] or

maximize the total throughput of the system [16]. Bit-level QoS constraints can also be included in the optimization problem to guarantee QoS. This QoS is expressed in terms of an upper bound on the bit error rate (BER), which can be translated into SINR threshold. The powers and rates should be allocated in such a way that the SINR of all the links be above the threshold. Similarly, rate guarantee can also be implemented to provide QoS based on rate requirements [21]. However, the optimal scheduling problem for peer-to-peer concurrent transmissions is NP-hard [22]. Since PNC itself is an ordinary device in terms of computational capacity, optimal scheduling would be a significant burden on it. To reduce the computation load, Liu *et al.* proposed two simple heuristic scheduling algorithms with polynomial time complexity [22]. In [23], Jiang *et al.* tried to solve this problem with an effective distributed resource allocation. Using this method, the burden of resource allocation is distributed among all the nodes. They have also proposed a novel message exchange procedure to convey control messages required for the distributed algorithm.

2.4.2 Scheduling Proposals for IEEE 802.15.3 Standard for MAC

The proposed channel time allocation algorithms for piconet generally consist of two parts: signaling and scheduling. As the central coordinator, PNC is responsible for scheduling. Depending on the scheduling algorithm, PNC requires certain information such as the number of flows, their reserved rates, their queues' status, type and deadline

of the frames in their queues, in order to allocate CTAs to different flows. Therefore, a signaling scheme is necessary so that DEVs can inform PNC of the required information. As specified in the standard, DEVs can use CAP for this purpose; however, this scheme has limitation in providing QoS for delay sensitive traffic. The devised frame format is also limited in the diversity of information that can be conveyed to the PNC. Hence, many proposed channel time allocation algorithms propose their own signalling scheme, so that the scheduler receives all the necessary information in a timely manner. In this subsection, we focus on the scheduling and signaling schemes for piconet.

General Schemes

It has been shown that the Shortest Remaining Processing Time (SRPT) scheduling algorithm minimizes the number of pending jobs in the system, and thus minimizes the average waiting time of the jobs [24]. It schedules different jobs in the system in the order of their remaining processing time, from the shortest to the longest. In the preemptive case, SRPT switches to a newly arrived job if the processing time of that job is shorter than the job currently being served. (In an 802.15.3 piconet, the CTA requirements of a flow can be regarded as a job.)

In systems with heterogeneous flows that have different average data rates, SRPT naturally favors the flows with smaller average data rate since they tend to have smaller frames. Motivated by merits of SRPT, Mangharam *et al.* proposed a solution to this natural unfairness of SRPT. In [1], they proposed a slight variation of SRPT, which

maintains fairness among flows with different data rates. This scheduling algorithm, called Fair-SRPT (F-SRPT), first normalizes the size of the frames belonging to a flow to the mean rate of that flow. This normalized size is then used by SRPT to sort the frames. As a result of the normalization in the first step, flows with smaller mean data rate will not dominate channel access. Hence, network resources are allocated in a fair manner.

The channel time allocation algorithm proposed in [1] guarantees that all the flows receive at least as much as their reserved amount of CTA. If a flow does not require all of its reserved CTAs (i.e., under-loaded), the excess is added to the idle channel time. This idle channel time is shared among the overloaded flows, which require more CTA than their reservation. It is highlighted in [1] that the subtle variations in allocation of idle channel time to overloaded flows can significantly affect the overall scheduling performance. Their simulation results show that using F-SRPT gives a lower JFR compared to the IEEE 802.15.3 standard.

The F-SRPT scheduling algorithm proposed in [1] includes a signaling scheme, called the *one-byte solution*. As the name suggests, a field with one byte is added to the MAC header. A DEV can use this field to inform the PNC of its current queue size. Despite its simplicity, this approach has a few drawbacks. First, this scheme works best when PNC is the destination of all the flows in the system, so that the PNC would be able to extract the extra one-byte from the MAC header and use it for scheduling. This is generally not the case, since PNC is not responsible for packet forwarding. Second,

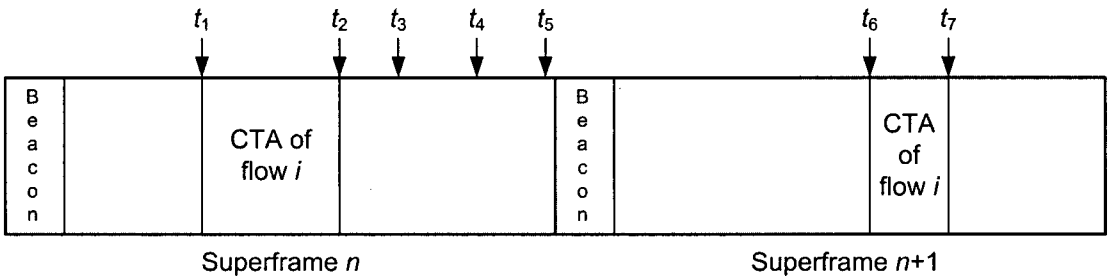

Figure 2.7: Flaws of one-byte solution for updating the status of the queues at the PNC [1]. In superframe n from time t_1 to t_2 is allocated to flow i. Using the one-byte approach, PNC has the queue status of flow i up to time t_2. At time t_3 and t_4, new data arrive for flow i and change its queue status. Therefore, PNC schedules the flows in superframe $(n+1)$ at time t_5 based on incorrect information about flow i. Flow i cannot update its queue status sooner than t_6.

the last opportunity for a DEV to update the PNC about its queue status is the end of its allocated CTA period (see Figure 2.7). Therefore, if new data frames arrive for one of the DEV's flows and change its queue status, it cannot update PNC sooner than its allocated CTA within the next superframe. As a result, PNC schedules the flows based on inaccurate information about the status of the DEV's queues. Lastly, most of the one-byte-information units sent to the PNC are not necessary and are not used by the scheduler. As far as the scheduling algorithm is concerned, the only useful piece of information is the most recent one, which is sent to the PNC in the header of the last MAC frame.

In [25], Liu *et al.* proposed a signaling scheme to mitigate the flaws of the one-byte solution. It allocates one MCTA to each flow at the end of the superframe. This MCTA is used to send control information to the PNC. The main advantage of this scheme is

that it does not alter the IEEE 802.15.3 standard. Furthermore, since the MCTAs are at the end of the superframe, more recent information is given to the PNC. This scheme reduces the average queueing delay of the system.

The optimality of SRPT, in that it minimizes the average waiting time of the jobs in the system, holds only when the jobs have no delay constraint. Thus, in the systems with delay sensitive traffic, SRPT need not be the best scheduler, and using deadline information can yield better performance. One scheduler that considers the deadline of the frames is the earliest due date (EDD). It schedules the frames in the order of their deadlines, from the earliest on. It is proven that EDD policy minimizes the probability of packet dropping in the system due to delay violation [26].

However, in wireless systems like WPANs, where the PNC should be informed of the internal state of the system queues, it is not practical to convey the exact deadline. This information is represented in a quantized form, usually with respect to the superframe size. For instance, a DEV may inform the PNC that the frame in its queue will expire after j full superframes. As a result of quantization, many frames can have equal deadlines, hence a tie-breaking method is needed.

In [2], Torok *et al.* used EDD along with SRPT as the tie-breaking method. They mainly studied the performance of this scheduler under real-time traffic with variable frame sizes. They modeled this traffic with batch arrival process. Each batch is the MAC fragments of a frame and all have the same deadline. In addition, the inter-arrival time of the batches is equal to the inter-arrival time of the frames and can be constant

or variable. For each flow, PNC has the information about the size and deadline of its first (head-of-the-queue) frame, as well as the total size of its queue. EDD+SRPT scheduler first selects the flows based on the deadline of the first frame in their queue (i.e., according to EDD). If the first frame of several flows have equal deadline, then they are scheduled in ascending order of their frame size (i.e., according to SRPT). After serving the frames closest to expiration, the remaining channel time, if any, is allocated to the flows according to their total queue size using SRPT algorithm.

A new signaling scheme is also presented in [2]. When control information is piggyback onto the data, it requires a complementary method to signal new arrival during the off period of data. Reserving fields in the MAC header for piggyback control information also incurs excessive overhead. The signaling method in [2] provides a solution. In this scheme, when a DEV finishes the transmission of its burst, a time slot is assigned to the DEV for sending its control packet. If the DEV has no data to send, the PNC allocates one time slot to the DEV in each superframe so that it can signal new arrival.

Reference [2] also describes *burst eligibility* method which improves the scheduling efficiency. Channel time wastage can be caused by partial transmission of bursts. When a burst expires before being completely transmitted, the transmitted portion is useless for the receiver and wastes channel time. Using burst eligibility scheme, the PNC determines if a burst can be fully transmitted by its deadline. If this is not possible, the burst is not scheduled at all. Thus, channel time can be utilized more efficiently.

Channel time wastage can also be caused by incomplete usage of CTAs. Allocating

the exact instantaneous required CTA to the flows in the system incurs a lot of overhead caused by the control messages that convey the status of the queues. Furthermore, the variable size of the frames makes it even more difficult to allocate the exact required CTA. Also, if a DEV does not receive a beacon correctly (because of channel error), it cannot use the CTAs allocated to it, and those CTAs would be unused.

An idle timeslot reuse scheme is proposed in [27] to increase channel utilization. This scheme consists of three phases: detection, cancelation and reuse. The detection of idle CTA can be done in two ways. First, the PNC can determine if a CTA is idle using clear channel assessment (CCA) procedure. Second, when a DEV partially uses its CTA, it may send a cancellation request to the PNC and let the PNC know that it is not using the rest of its CTA. After detecting the idle CTA, the PNC broadcasts a cancellation message. After that, all the DEVs can contend to use the idle part of the CTA. The result is higher channel utilization and lower average queueing delay.

According to the current IEEE 802.15.3 standard, once a CTA request is received from a DEV, the PNC shall remember that as the outstanding request for that stream for every superframe until another CTA request for that stream is received from the DEV. Since CTAP is shared between different priority classes, it can fill up very fast, and new connections will be rejected regardless of their priority class. This can cause undesirably long waiting time and poor QoS for high priority streams. A QoS degradation policy is proposed in [28] to mitigate this problem. It is suggested that the lower priority traffic lend some channel time to the newly arrived traffic with higher priority so that they

do not get rejected. An $M/M/c/\infty$ queueing model is used to analyze the impact of changing c (number of available CTAs) on the average waiting time. By getting some share of low priority traffic in favor of the higher priority class, some CTAs are released (i.e., c increases), and consequently the average waiting time reduces.

As a result of reducing the average waiting time, more DEVs get the chance to use CTAP. This causes more CTA requests and other commands to be transmitted during CAP, and the limited capacity of CAP may partly cancel out the benefit of the proposed scheduling algorithm for CTAP. In order to increase the number of successful requests (or other commands in general) during CAP, the overhead should be decreased. The command frames have very long headers and preamble, compared to their actual payload. The command-aggregation scheme proposed in [28] divides CAP into two parts: CAP1 and CAP2. During CAP1, each DEV aggregates all of its commands in a single command and sends it to the PNC within one access to the channel. During CAP2, the PNC aggregates the responses to all the commands, again in a single message. Aggregating the commands significantly reduces the overhead of headers and preamble. It also reduces the number of contentions within CAP. Thus, the collision probability is decreased. The result is the overhead reduction and capacity improvement of the CAP period, so that the enhancement in the CTAP can be fully utilized.

Application-aware Schemes

In this section, we summarize the scheduling proposals that are specifically designed to provide QoS to video streams. They use application layer information for scheduling video frames, namely the type of the frames.

Kim and Cho suggested that each video frame type be scheduled with a pre-assigned priority (PAP) based on its importance, i.e., in I, P, and B order [3]. EDD and SRPT are used to break ties among frames of the same type, when they cannot be served together in one superframe.

They also proposed using mini packets with short duration for signaling purposes. This method divides the CTAP part of the superframe into two parts. The first part is CTA and is used for transmission of real-time traffic. The second part is Feedback CTAs (FCTAs). The PNC allocates this part for transmission of mini packets.

There are two classes of information contained in the mini packets. The first class is used for CTA allocation to video flows in the system. This includes data rate, ACK policy and fragmentation threshold between source and destination of the flow, as well as video frame type and queue status information of the source of the flow. These pieces of information are contained in the mini packet only when necessary (i.e., when their value changes). The second class of information is used for deciding the allocation of FCTA for mini packets. In order not to waste channel time with unused FCTA, it is only allocated to the DEVs that need to send mini packet to the PNC (i.e., when a new video frame arrives at the DEV). Therefore, this class of information includes the time of the

next arrival. By using that information, the PNC can allocate FCTAs only to the flows that need it. After receiving all the mini packets from the DEVs, the PNC performs scheduling based on the received information.

In the scheme proposed in [29], each DEV informs the PNC about the maximum size of its I, P and B frames, as well as its GOP pattern. Thus, the PNC can determine the type of each frame, and allocates channel time to it as much as its maximum size. Since even one frame type of video stream has high maximum to mean size ratio, this scheme suffers from low channel utilization.

Energy efficiency is considered by the scheduler proposed in [30]. It defines different service categories in order to balance between energy efficiency and QoS. It also uses the application layer information to assign priorities to the buffered frames at the source of a flow. Furthermore, a simple flow admission control is proposed to admit new flows of different service categories to the system.

Other Schemes

Choosing a proper ACK policy for data communication is not addressed in the IEEE 802.15.3 standard for MAC. In [31], Xiao *et al.* studied optimal ACK policies under UWB channel error condition. For a given ACK policy and error channel condition, they analytically determined the optimal payload size, which maximizes the throughput. Since CTAP and CAP have different channel access mechanism, the optimal ACK policies are studied separately for each case. In CTAP, the analytical model gives the optimal

payload size for each ACK policy, given a fixed (allocated) CTA. In CAP, however, a model similar to Bianchi's model [32] is used to calculate CAP throughput for each ACK policy. The optimal payload size is calculated by maximizing the throughput.

The proper burst size of Dly-ACK policy is also not covered in the standard. The overhead of Imm-ACK policy is very high because ACK frame is sent at the base rate, which can be much less than the data rate. Furthermore, the interframe spacing of Imm-ACK (SIFS) is longer that that of Dly-ACK (MIFS). Therefore, using Dly-ACK can potentially reduce the overhead caused by ACK frames. Liu *et al.* [33] present an adaptive delay acknowledgement algorithm for video traffic. It dynamically adjusts the burst size of the Dly-ACK policy in response to the channel quality. The results show that this scheme gives a higher throughput and lower JFR, compared to Dly-ACK with fixed burst size or Imm-ACK.

2.5 Markov Decision Process (MDP)

In real life, as well as laboratory and simulation settings, there are many cases where one should make decisions while accounting for the relationship between present and future decisions. In other words, decisions are not made in isolation. Such decision making problems can formally be analyzed using the model for probabilistic sequential decision making under uncertainty. In this model, a decision maker or an agent observes the system state at any decision making epoch, based on which it chooses an action. As a result of the chosen action, the agent receives (incurs) an immediate reward (cost)

and the system evolves to a new state according to a probability distribution. Both the reward and the transition probabilities depend on the state and the chosen action. At the next decision epoch, the agent faces a similar decision making problem. However, the system may be in a different state and there may be a different set of actions to choose from [34].

At any decision epoch, the action is chosen according to a *decision rule*, which may depend on the current state alone, or on all previous states and actions. A *policy* provides the agent with a set of decision rules so that the agent can choose an action in any possible system state. In other words, the agent uses the policy as a prescription to choose an action in any situation. When an agent implements a policy, it receives a sequence of rewards over time. The sequential decision problem is: given the system information (i.e., the set of decision epochs, states, actions, rewards and transition probabilities), what the optimal policy is with respect to a predetermined performance criterion. This criterion can be a function of the reward sequence yielded by the policy. Possible choices for this function include the expected total discounted reward or the long-run average reward [34].

Markov decision process (MDP) is a special case of sequential decision making model, where the set of available actions, the rewards, and the transition probabilities depend only on the current state and action, and not on the visited states and chosen actions in the past. Any decision making problem modeled by MDP has the following elements: decision epochs, system states, available actions, transition probabilities, and immediate

rewards.

Decision epochs are the points of time at which the agent makes decisions. The set of decision epochs can be either a discrete set or a continuum. It can also be classified into finite or infinite set, based on its size. The MDPs based on the former are finite horizon, as opposed to infinite horizon MDPs that are based on the latter. In this chapter, we consider the discrete set of decision epochs for infinite horizon MDPs. Thus, the set of decision epochs is the set of integers. At each decision epoch, the system is in a *state*. This state describes the status of the system at that epoch. We denote the set of possible states by S. Hence, the underlying Markov chain of an MDP with discrete epochs set is given by $\boldsymbol{X} = \{X_n : X_n \in S,\ n = 0, 1, \dots\}$, where X_n denotes the system state at the nth decision epoch. Depending on the system state $s \in S$ at any decision epoch, there is a set of available actions A_s that the agent can choose from. When the agent chooses an action $a \in A_s$, the system state evolves according to the transition probability $p(s'|s, a)$, which represents the probability of moving from state s to state s' under action a in one step. Furthermore, the agent receives an immediate reward $r(s, a)$. This reward function is defined as r: $S \times A \rightarrow \mathbb{R}$, where $\mathbb{R}$ denotes the set of real numbers, and $r(s, a)$ is the expected reward for taking action a in state s. That is, $r(s, a) = \sum_{s' \in S} r(s, a, s')p(s'|s, a)$, where $r(s, a, s')$ is the immediate reward in transition from state s to state s' under the action a.

Sequential decision problem consists of stages of decision making and receiving reward. This allows the use of of dynamic programming, where the whole multi-stage problem

of finding the optimal value can be reduced to a sequence of simpler inductively defined single-stage "choosing action and receiving reward" problems.

When the decision making agent follows policy π up to the decision epoch T, it yields a bivariate discrete-time reward process given by [34]

$$\{(X_n, r(X_n, Y_n)) \mid X_n \in S, Y_n \in A_{X_n},\ n = 0, 1, \ldots, T\}.$$

The first component is the state of the system at decision epoch n which was described before. The second component is the reward received when action Y_n is chosen in state X_n according to policy π. The *average reward* or *gain* of policy π is defined to be

$$\rho^{\pi}(s) \triangleq \lim_{T\to\infty} \frac{1}{T} E_s^{\pi}\left\{\sum_{n=1}^{T} r(X_n, Y_n)\right\}$$

where $E_s^{\pi}\{\cdot\}$ is the expectation over all the possible state-action sequences beginning with s that the policy can yield: $\{(X_1, Y_1, X_2, Y_2, \cdots) \mid X_1 = s\}$. Dynamic programming (DP) methods can find a stationary deterministic policy π^* (which is a mapping π^*: $S \rightarrow A$), that is optimal with respect to the average reward criterion (i.e., optimal gain policy). *Value iteration* and *modified value iteration* [34] are examples of DP technique.

2.6 Reinforcement Learning (RL)

In many real-life sequential decision making problems, DP techniques are not practical because of two main reasons. First, the agent may not have complete knowledge about dynamics of the system or environment. Second, when the system state and/or action space is huge (e.g., continuous state spaces), it is not feasible to exploit DP. The

former problem is called *curse of modeling*, while the latter is called *curse of dimensionality*. Reinforcement learning (RL) combines DP methods with other methods including stochastic approximation and function approximation to tackle these problems [35]. RL is the problem faced by a decision making agent that must learn from its own experience of interacting with a dynamic environment. The agent should learn a policy (i.e., how to map situations to actions) through trial and error. The agent is not provided with the policy, but instead it must discover which actions yield the most reward by trying them. Figure 2.8 illustrates the standard RL model. In this model, the agent has an indication i of the system state s (e.g., if the agent is a robot, this indication can be the signals that the robot receives from its different sensors). The agent takes action a, which leads the system to another state. The value of this state transition is communicated to the agent through a scalar reinforcement signal (or reward) r. The agent should choose actions that tend to increase some function of r. Similar to DP, this function can be average discounted reward, average reward, or other reward functions. Unlike DP, however, the agent should learn to maximize this function over time by systematic trial and error, guided by a wide variety of algorithms [35][36]. The system environment can be either real or simulated. If a simulator is used, complete knowledge of the random variables that govern the system dynamics is required [37].

In order to obtain a lot of rewards, the agent should be greedy; i.e., it should choose the actions that it has discovered to produce a lot of reward. In other words, it should *exploit* its current experience. However, there might exist better actions that the agent

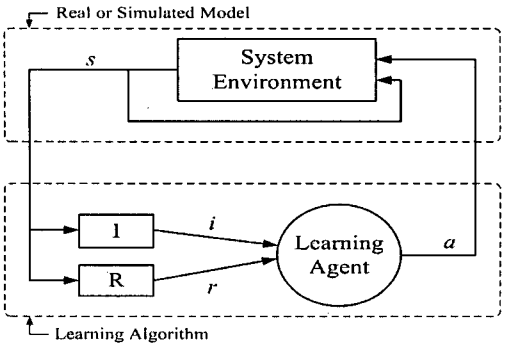

Figure 2.8: Standard RL model

has not yet discovered or explored. As a result, if the agent occasionally deviate from the greedy manner (i.e., perform *exploration*), chances are that it can find better actions. The dilemma in RL is that neither exploration nor exploitation should be pursued exclusively. In fact, one of the challenges of RL is to balance the trade-off between exploitation and exploration. The agent must try a variety of actions and progressively favor those that appear to be best. On a stochastic task, each action must be tried many times to gain a reliable estimate of its expected reward [36]. In the next section, we elaborate more on how RL overcomes the challenges that render DP to be impractical.

2.6.1 Curse of Modeling

As mentioned in Section 2.5, DP requires the exact information about all the components of the MDP that it tries to solve. The component that gives the system model is the transition probabilities $p(s'|s,a)$ for all states and actions. In many applications, this model is not available. In this section, we summarize an algorithm to overcome this

problem, which is proposed in [38].

Model-free Q-Learning

When the agent does not have the system model, it should build a knowledge base according to its interaction with the system. In Q-learning, this knowledge base is made up of factors called Q-values. For each state $s \in S$ and each action $a \in A_s$, the agent saves a Q-value $Q(s,a)$ and updates these values as the agent learns. Before the learning begins, all the Q-values are set to the same value. At a decision epoch, when the system is in state s, the agent chooses an action a that maximizes the Q-value (i.e., $a = \arg\max_{a' \in A_s} Q(s,a')$). The simulator will then simulate the chosen action a in the current state s and evolve to the new state s'. It also gives the accrued reward and the time spent during the state transition. The learning algorithm embedded in the agent will then use this information to update $Q(s,a)$. Basically, $Q(s,a)$ increases if a was a good action (i.e., good actions are rewarded), and decreases otherwise (i.e., bad actions are punished). As mentioned before, an exclusively greedy learning algorithm is not the best. Consequently, the agent should occasionally explore and choose an action other than the one with the maximum Q-value. Over time, as the agent explores and tries out all the state-action pairs often enough, it learns the best action in each state. In other words, it learns an optimal (or near-optimal) policy. At this point, the agent should cease exploration and follow the learned optimal policy in a greedy manner.

Q-learning is a value iteration based RL. It learns the Q-value of taking action a in

state s using the following updating procedure [37]:

$$\begin{aligned} Q(s,a) &= r(s,a) + \sum_{s' \in S} p(s'|s,a) \max_{a' \in A_{s'}} Q(s',a') \\ &= \sum_{s' \in S} p(s'|s,a) \left[r(s,a,s') + \max_{a' \in A_{s'}} Q(s',a') \right] \end{aligned} \tag{2.1}$$

where the last equality follows from the definition of $r(s,a)$. There are at least two reasons that RL, including Q-learning, requires the updating in equation (2.1) to be gradual [37]. First, RL is asynchronous (i.e., unlike DP, the Q-factors of state-action pairs are not all updated simultaneously at each iteration) and requires a proper learning rate to converge. Second, the major goal of RL is to avoid computation of transition probabilities, which also requires a learning rate. For a learning rate $\mu \in (0,1]$, we can write the equation (2.1) as:

$$Q(s,a) = (1-\mu)Q(s,a) + \mu \sum_{s' \in S} p(s'|s,a) \left[r(s,a,s') + \max_{a' \in A_{s'}} Q(s',a') \right] \tag{2.2}$$

Since equation (2.2) uses the probability distribution $p(s'|s,a)$, it gives the updating equation for *model-based* Q-Learning algorithm, which still suffers from the curse of modeling. Stochastic approximation schemes can be used to circumvent this problem. One such scheme is proposed by Robbins and Monro [39], which is used to approximate the mean of a random variable with methods such as simulation. Given a stochastic function $f(\mathcal{Q})$, the updating procedure $\mathcal{Q} \leftarrow (1-\mu)\mathcal{Q} + \mu[f(\mathcal{Q})]$ with a suitable learning rate produces an averaging effect (i.e., converges to the average of $f(\mathcal{Q})$) if updating is performed frequently enough. This updating procedure is very similar to equation (2.2). Since the term $\left[r(s,a,s') + \max_{a' \in A_{s'}} Q(s',a')\right]$ is averaged anyway, one can replace the

expectation over s' in equation (2.2) by a *sample*. Thus, the Robbins-Monro version of Q-learning can be written as follows [40]:

$$Q(s,a) \leftarrow (1-\mu)Q(s,a) + \mu\left[r(s,a,s') + \max_{a' \in A_{s'}} Q(s',a')\right] \tag{2.3}$$

which is free of transition probabilities. Equation (2.3) describes the *model-free* counterpart of equation (2.1). Since the average reward value iteration can be numerically unstable [34], the numerically stable version of model-free Q-learning should be based on relative value iteration. We describe this in the next section.

SMART Algorithm

Most of the RL algorithms in the literature, including Q-learning, are based on the discounted reward optimality criterion. Moreover, these algorithms cannot automatically extend to the average reward criterion [41]. As we will show later in Section 4.2.5, the average reward criterion is more suitable for our scheduling problem. We use an algorithm called SMART (Semi-Markov Average Reward Technique) [38][42][40] that is designed for average reward RL. The convergence analysis of this algorithm is given in [42] and it has been successfully applied to different problems including production inventory [38], airline seat allocation [40], and QoS provisioning in wireless cellular networks [41]. The difference between Semi-MDP (SMDP) and MDP is that for SMDP, the time that it takes the system to transit from one state to the other is stochastic. MDP is a special case of SMDP when the state transition duration is fixed. Since in Section 4.2 we formulate the scheduling problem in the form of an MDP, we can use this algorithm with fixed state

transition duration.

SMART is an RL version of relative value iteration algorithm [34] and subtracts the average reward in each update. The MDP version of the update procedure for SMART is as follows [38]:

$$\begin{aligned} Q(s,a) &\leftarrow (1-\mu)Q(s,a) + \mu\left(r(s,a,s') - \eta\rho + \max_{a'} Q(s',a')\right) \\ &= Q(s,a) + \mu\left(\left[r(s,a,s') - \eta\rho + \max_{a'} Q(s',a')\right] - Q(s,a)\right) \end{aligned} \tag{2.4}$$

where η is the fixed state transition time, ρ is the average reward (see Section 4.3.3), and μ is the learning rate. SMART is the base of our scheduler proposed in Chapter 4.

2.6.2 Curse of Dimensionality

Besides a lack of knowledge about the system model, the agent may also face a system with vast state and/or action space to interact with. In this section, we address the measures for tackling this challenge.

State Aggregation and Features

In the systems with large state space, it is sometimes possible to aggregate similar states with slight degradation in accuracy. For instance, consider a problem where a robot needs to learn how to find a moving object in a rectangular 2-D plane. The obvious choice of state space is the coordinates of the robot and the moving object. However, this state space has continuous variables and its size is not finite. To use aggregation in

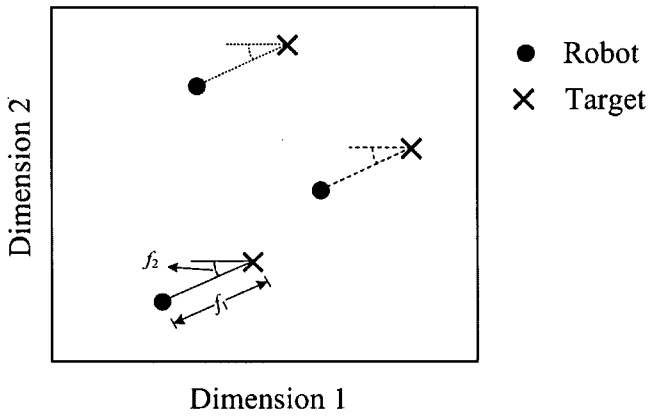

Figure 2.9: This figure shows a 2-D plane, in which a robot tries to find a target. Robot and target are depicted in three different positions, which are all equivalent with respect to the value of features f_1 and f_2. These different states can be aggregated.

this problem, the 2-D plane can be divided to small rectangular tiles. The center of each tile can then represent the coordinates of the robot or the target when they are in that tile. Thus, all the points of the state space that lie inside a tile are aggregated to the center of the tile.

Depending on the learning problem, there might exist some *features* in the states that can fully capture the important aspects of system that influence the learning process. For instance, consider the same problem as in the previous paragraph, where the state space is the coordinates of the robot and the moving object. Using the concepts of simple geometry, it can be verified that as far as the target finding problem is concerned, this state space only has two important features: f_1, the distance between the robot and the target, and f_2, the angle between the robot-target line and the horizontal line passing from the target (see Figure 2.9). Since all the points in the state space that yield the

same value for features (f_1, f_2) can be aggregated and represented by that value, features can be viewed as a form of aggregation. In general, the choice of suitable features is not a straightforward task, and it requires good understanding of the problem, intuition, and experimentation. For large discrete state spaces, state aggregation can reduce the memory requirement of look-up tables and make it feasible to use. However, as we will describe in the next section, memory requirement is not the only challenge in problems with large (or continuous) state spaces. Therefore, state aggregation, by itself, may not solve the curse of dimensionality. Note that in the case of Q-learning, where the agent deals with state-action pairs, aggregation should be used in the state-action space.

Generalization and Function Approximation

In Q-learning, the knowledge base includes a Q-value for each state-action pair. This is only practical for problems with small state and action spaces. One reason is that the memory requirement for the lookup table is $O(|S| \cdot |A|)$, which grows very large for many practical problems. In addition, the lookup table method cannot be directly applied to the problems with continuous state and/or action spaces.

Besides memory requirement, another problem with large lookup tables is the amount of time required to fill them up accurately [36] because lookup tables use the gained experience inefficiently. For instance, lookup table does not take advantage of the fact that in a large smooth state space, we generally expect similar states to have similar optimal actions [35]. In many tasks, most encountered states have never been tried

exactly before. This is usually the case in complex problems or when the state or action spaces include continuous variables [36]. As a result, a more compact representation that allows transfer of experience between similar state-action pairs should be used. In other words, this compact form should *generalize* from previously experienced state-action pairs to the ones that have never been experienced. The kind of generalization that is required in Q-learning is often called function approximation, because it tries to generalize the Q-values that are experienced to approximate the entire Q-function.

Generalization techniques that are successfully applied to RL include artificial neural network methods [43], fuzzy logic [44], cerebellar-based (CMACs) or tile coding [45][46], and memory-based methods [47].

A class of function approximators called sparse coarse-coded function approximators nicely provides generalization with reasonable memory requirement [48]. This class is the most basic model of associative memory, where several memory locations are used to give the content associated with an input. This value would be similar for two different inputs, if they share many memory locations. In fact, generalization is the result of inputs sharing memory locations. Tile coding and memory-based techniques fall in this class.

In memory-based techniques, each memory location (or prototype) corresponds to a group of state-action pairs that the agent has experienced before. In other words, each prototype covers a region of state-action space and is activated by any point in that region. Furthermore, each state-action pair can activate more than one prototype. The set of prototypes can be fixed (e.g., uniformly distributed in the state-action space), or

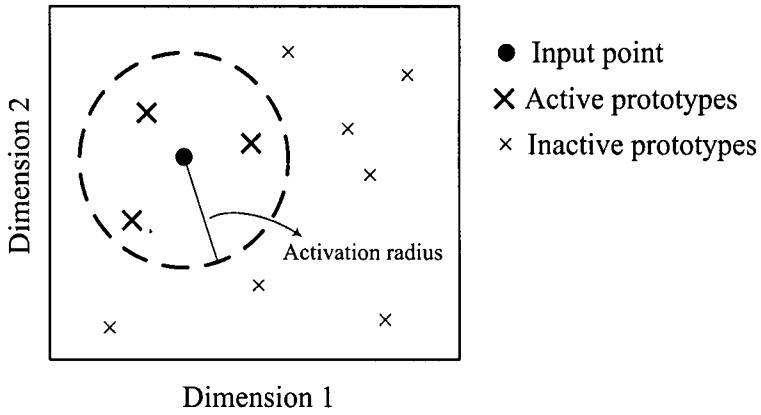

Figure 2.10: An example of memory-based function approximator. The input point is the point in the state-action space that its Q-value is desired. All the prototypes within the activation radius of the input are activated. The weighted sum of the values of the activated prototypes is the approximated value of the input.

dynamically expand as the agent explores new regions of the state-action space. One method for dynamic expansion of prototypes is to add enough new prototypes in order to maintain a minimum density of prototypes along the explored trajectory.

In the case of Q-learning, the prototypes are used to approximate the entire Q-function. When the agent tries a new state-action pair, its Q-value is calculated as the weighted sum of the prototypes' values (i.e., contents of the memory locations) that are activated by this pair. In order to determine what prototypes are activated by an input, a similarity metric is required. For instance, in Figure 2.10, all the prototypes within an activation radius of an input are similar to and activated by it. In Chapter 4, we use a memory-based function approximator for our proposed scheduler.

2.7 Summary

In this chapter, we first introduced different physical layers based on UWB transmission, which enable WPANs to support high-rate applications, such as video streaming. We also investigated the characteristics of video streams, as one of the major traffic types in future high-rate WPANs. It was noted that for video flows, which have inter-frame dependencies, it is more appropriate to use DFR as the performance metric for scheduling. In addition, we described the related work on scheduling at MAC layer, many of which were based on the impulse-based UWB and took advantage of concurrent transmission allowed by this type of physical layer. A few application-aware scheduling proposals, which use application layer information for scheduling video frames, were also mentioned. However, the related work on this class of MAC schedulers is limited. Furthermore, the current proposals do not use the application layer information effectively. An introduction to MDP, challenges of solving large-scale MDPs, and RL were also presented.

In Chapter 3, we propose a new technique for video schedulers to incorporate application-layer information more efficiently. We show that this simple technique can significantly improve the scheduling performance measured in terms of DFR.

Chapter 3

Frame Decodability Aware (FDA) Technique

In this chapter, we propose a method to improve video traffic schedulers in IEEE 802.15.3 WPANs, called the frame decodability aware (FDA) technique [6]. We first state the motivations and assumptions, followed by the description of our FDA technique. We then present the simulation results for the impact of the FDA technique on several scheduling algorithms, namely F-SRPT [1], EDD+SRPT [2], and PAP [3] algorithms.

3.1 Motivations and Assumptions

Similar to some of the previous work described in Chapter 2, our proposed scheme uses information such as frame size, deadline, and type of the frame. However, one can also determine the decodability of the queued frames in the system at no extra signalling cost. To our knowledge, none of the previous work have explored this opportunity, which is readily available at the PNC, for improving scheduling performance under video traffic. This is the main motivation of our proposed FDA technique, which will be described in the next section. This technique has the following properties:

- It provides a simple way of exploiting the information about video frame type, hierarchical structure, and decoding dependencies in order to improve the scheduling performance. As in [3] and [49], we assume there is enough cross-layer interaction to provide the information about frame type at the MAC layer.

- It is independent of the scheduling algorithm implementation, which is important from design point of view and allows more flexibility.

For simplicity, we assume that the transmission deadline of video frames is constant, and is equal to the frame inter-arrival time. Thus, at any given time, the queue of each flow holds no more than one video frame. We also assume that the length of superframe is a constant. In general, since the size of a video frame is larger than the maximum size of the data frame passed from the upper layers, fragmentation is performed at the MAC layer.

3.2 FDA Technique

Let $\mathcal{F}$ denote the set of flows. In FDA, the PNC creates a *scheduling eligibility table.* The number of entries is equal to the number of flows $F = |\mathcal{F}|$. The entry for flow $i \in \mathcal{F}$ includes the following information:

- The remaining amount of channel time l_i that should be allocated to flow i;

- The deadline d_i, no later than which l_i should be allocated to flow i;

- The type of the video frame in the queue of flow i, which is denoted by y_i.

The unit of l_i is in milliseconds and is locally determined by the DEV to which flow i belongs. It depends on the length of the video frame, as well as the ACK policy and channel rate used by the source and destination of the flow [5]. Since the length of superframe is constant, the unit of d_i is in terms of the number of superframes left until the deadline expires. The type of a video frame $y_i \in \{I, B, P\}$.

The FDA technique determines the decodability of each frame based on the deadline and type information. For each flow $i \in \mathcal{F}$, we define a binary variable δ_i. The value of δ_i is set to 0 if any I or P frame on which the current frame of flow i depends is directly lost; otherwise, δ_i is set to 1. According to the hierarchical structure of video encoding, a frame depends on its preceding I and P frames up to the most recent I frame. Therefore, when $y_i = I$ or P, δ_i indicates whether the frame of flow i is decodable, (i.e. neither directly nor indirectly lost). If δ_i is equal to 0, the subsequent frames up to the next I frame (i.e., beginning of the next GOP) are indirectly lost. However, when $y_i = B$, δ_i only indicates that the frame of flow i is not indirectly lost. The entry of each flow also includes a variable e_i, which shows whether or not the frame of flow $i \in \mathcal{F}$ is eligible for being scheduled.

The PNC updates the scheduling eligibility table at the beginning of each superframe before making the scheduling decisions. The pseudo-code of the table update for entry i is shown in Figure 3.1. The table is updated by the following procedures: If some channel time is allocated to flow $i \in \mathcal{F}$ in the previous superframe, then l_i is reduced

For each flow $i \in \mathcal{F}$, the scheduling eligibility table maintains the following info:

- l_i is the remaining amount of channel time that should be allocated to flow i.
- d_i is the deadline of allocating l_i to flow i.
- y_i is the type of the video frame in the queue of flow i, $y_i \in \{I, P, B\}$.
- δ_i is set to 0 if any I or P frame, on which the current frame of flow i depends, is directly lost; and is equal to 1 otherwise.
- e_i indicates whether the current frame in the queue of flow i is eligible for being scheduled.

(A) At the beginning of each superframe, perform the following steps for each flow $i \in \mathcal{F}$:
$l_i \leftarrow l_i -$ (time allocated to flow i in the previous superframe);
$d_i \leftarrow (d_i - 1)$;
y_i is not changed;

if $(d_i = 0)$ and $(l_i > 0)$ **then**
if $(y_i = I)$ or $(y_i = P)$, **then** $\delta_i \leftarrow 0$;
else δ_i is not changed.

if the DEV corresponding to flow i has sent information about a new arrival in the previous superframe, **then**
$l_i \leftarrow l_i^{new}$; $d_i \leftarrow d_i^{new}$; $y_i \leftarrow y_i^{new}$;
where, l_i^{new}, d_i^{new} and y_i^{new} denote the required CTA, deadline and type of the new frame, respectively. This information is contained in the message sent to the PNC from the DEV.
if $y_i^{new} = I$, **then** $\delta_i \leftarrow 1$;
else δ_i is not changed.

if $(l_i > 0)$ and $(d_i > 0)$ and $(\delta_i = 1)$,
then set $e_i = 1$ (eligible for being scheduled);
else set $e_i = 0$ (ineligible for being scheduled).

(B) After updating the scheduling eligibility table, feed all the eligible flows to the scheduler.

Figure 3.1: Pseudo-code of the FDA technique.

by that amount. The value of l_i becomes 0 when the channel time request of flow i is fully accommodated (i.e., when the queue of flow i becomes empty). In any case, d_i is reduced by one in order to indicate that one superframe has elapsed from the deadline. If d_i reaches 0 and the queue of flow i is not empty ($l_i > 0$), then the frame in that queue has expired and is no longer eligible for transmission. This would impact the decodability of the subsequent frames of flow i if y_i is either I or P. In either case, we set δ_i to 0 in order to indicate that an I or P frame on which the upcoming frames of the current GOP depend is directly lost. Note that if the expired frame is of B type, then δ_i is not changed. The reason is that the decodability of subsequent frames is not affected by the loss of a B frame.

In case of new frame arrivals, the PNC should also update the entries of the flows that have new frames according to the information sent from the corresponding DEVs. The parameters l_i, d_i and y_i are set to the required CTA, deadline and type of the new frame arrived at flow i, respectively. If the new frame is of type I, then a new GOP has begun. The decodability status of the frames preceding the new I frame does not affect the decodability of the I frame and its subsequent frames. Therefore, when a new I frame arrives for flow i, we set δ_i to 1 regardless of the previous value of δ_i.

Finally, after updating the values of l_i, d_i, y_i and δ_i, the FDA technique determines if the frame of flow i is eligible for being scheduled. The frame is marked as eligible (i.e., $e_i = 1$) if the following three conditions are valid: (1) it is not completely transmitted ($l_i > 0$), (2) the frame has not expired ($d_i > 0$), and (3) the frame is not indirectly lost

($\delta_i = 1$).

When the scheduling eligibility table has been updated by FDA, the scheduling algorithm will be applied to the eligible frames. Note that the FDA technique is independent of the type of scheduling algorithm being invoked. It only gives the list of eligible frames to the scheduler. By using the FDA technique, the scheduler does not need to allocate channel time to those frames that are indirectly lost. As a result, some CTAs are freed up for scheduling other frames that are decodable at the receiver. In Section 3.3.2, we study the impact of the FDA technique on several scheduling algorithms.

Figure 3.2 shows an example of updating the scheduling eligibility table with 5 video flows in the piconet. At the beginning of superframe n (see Figure 3.2(a)), the frames of flows 3 and 4 expire. The flows 1, 2 and 5 are eligible for being scheduled. Suppose that in superframe n, 15 ms and 32 ms of channel time is allocated to flows 2 and 5, respectively. At the beginning of the next superframe $n + 1$ (see Figure 3.2(b)), the

i	l_i	d_i	y_i	δ_i	e_i
1	18	1	B	1	1
2	23	1	P	1	1
3	28	0	I	0	0
4	14	0	B	0	0
5	32	2	P	1	1

(a)

⇨

i	l_i	d_i	y_i	δ_i	e_i
1	18	0	B	1	0
2	8	0	P	0	0
3	21	3	B	0	0
4	37	3	I	1	1
5	0	1	P	1	0

(b)

Figure 3.2: An illustration of how the FDA technique updates the scheduling eligibility table. The table after being updated at the beginning of (a) superframe n, (b) superframe $n + 1$.

frame of flow 1 expires. Since this frame is of type B, it does not affect δ_1. The frame of flow 2 also expires and is directly lost. Since this frame is of type P, δ_2 becomes 0. A new frame arrives for flow 3 with CTA requirement, deadline and type equal to 21 ms, 3 and B, respectively. This frame is indirectly lost because its previous I frame is directly lost, as indicated by the value of δ_3. A new I frame arrives at flow 4. The value of δ_4 will be set to 1. Finally, the queue of flow 5 becomes empty as its channel time request is completely granted in superframe n. The frame of flow 4 is the only frame eligible for being scheduled in superframe $(n+1)$. Although $d_3 > 0$, the frame of flow 3 is not eligible for being scheduled because it is indirectly lost. By preventing the undecodable frames being scheduled, the FDA technique improves the utilization of the channel time.

The implementation of FDA requires a signaling scheme to pass the required info from DEVs to the PNC. For example, FDA can use the signaling method proposed in [25] which is compatible with the 802.15.3 standard. It allocates one MCTA to each flow, at the end of the superframe. Compared to using CAP or allocating MCTAs at the beginning of CTAP, this scheme gives more recent information to the PNC and reduces the average response time of the system. It is important to note that FDA does not depend on (or affect) the implementation details of the scheduler. It merely modifies the input to the scheduler (see Figure 3.3). Thus, FDA technique can be used in any scheduler to provide information about the decodability of video frames.

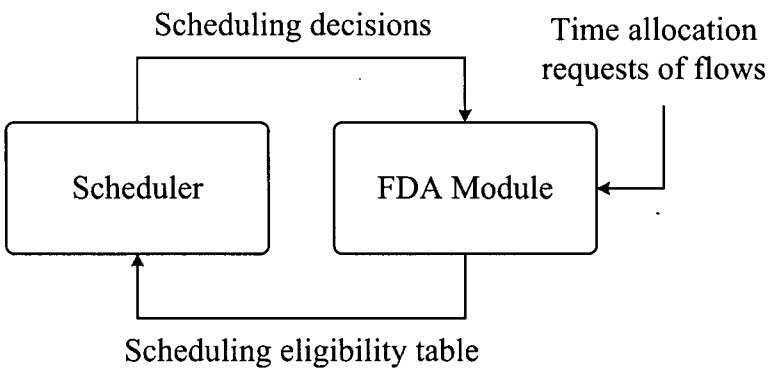

Figure 3.3: Logical interaction between FDA module and the scheduler.

3.3 Performance Evaluation

In this section, we evaluate the performance of various scheduling algorithms with (and without) the use of the FDA technique. The scheduling algorithms that we studied are the F-SRPT [1], EDD+SRPT [2], and PAP [3] algorithms.

3.3.1 Simulation Model

The implementation of our proposed FDA technique is done in MATLAB. We use the real video trace of *Jurassic Park* movie with GOP pattern of (N=12, M=3) [50]. The frame rate is 30 frames per second. The average rate of each flow is 8 Mbps. We assume that the maximum tolerable delay for video frames is equal to the frame inter-arrival time, which is 1/30 s. The number of flows vary from 2 to 10. The MAC overheads such as headers and inter-frame spacings are set as specified in the IEEE 802.15.3 standard [5]. The channel data-rate is equal to 100 Mbps, the superframe size is 8 ms, and the simulation time is equal to 500 s. Table 3.1 summarizes the simulation parameters.

Table 3.1: Simulation Parameters

Parameter	Value
Channel data rate	100 Mbps
Superframe duration	8 ms
Number of video flows	2 – 10
Video GOP pattern	(12, 3)
Mean data rate of video flows	8 Mbps
Frame inter-arrival time	1/30 s
Maximum tolerable delay	1/30 s
ACK policy	Imm-ACK
Max MAC fragment size	2048 bytes
Simulation duration	500 s

3.3.2 Simulation Results

The start time of different flows in the system affects the burstiness of traffic load, and thus influences the overall performance. In order to show this fact, we assume that there is ϕ seconds between the start times of successive flows. In other words, flow i starts at time $i\phi$ [1]. Figure 3.4 compares the average decoding failure rate (DFR) and job failure rate (JFR) achieved by F-SRPT and EDD+SRPT scheduling algorithms when ϕ varies from 0 to 30 ms, and the number of video flows is fixed to 9. In our simulation model, for small values of ϕ the traffic peaks of each flow coincides with that of other flows. Therefore, the overall load becomes more bursty and both JFR and DFR increase significantly. Thus, we conclude that ϕ can affect the performance. Another observation

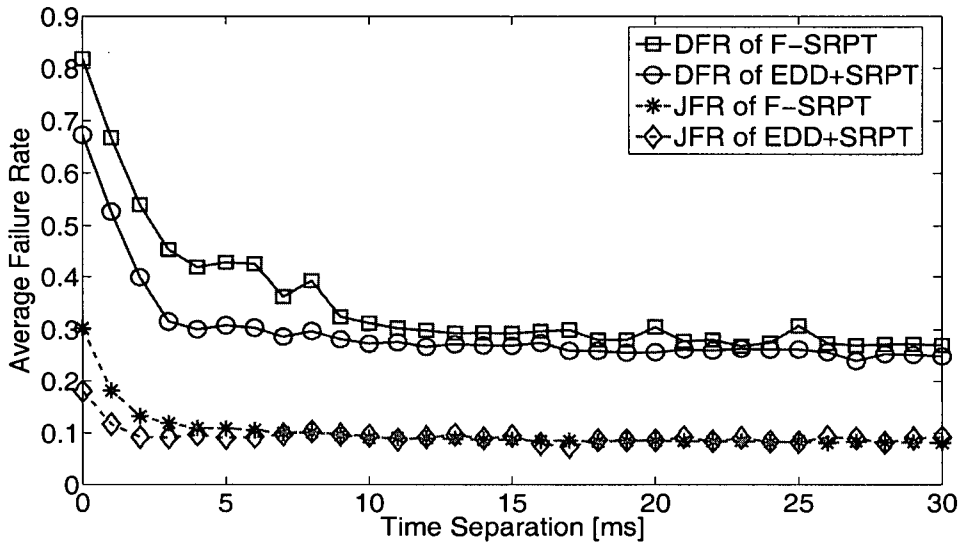

Figure 3.4: The effect of time separation ϕ on failure rate ($F = 9$).

in Figure 3.4 is the large gap between DFR and JFR for both schedulers. For most values of ϕ, DFR is more than 3 times as much as JFR. This validates our claim in Chapter 2 that for video flows, low JFR is not an accurate indicator of good QoS.

As mentioned in Section 2.3.1, B frames are likely to be smaller that I and P frames, so they are favored by the SRPT scheduler. Similarly, P frames have a better chance to be scheduled, compared to I frames. In other words, SRPT functions in favor of less important frames. As a result, many of the completed jobs (i.e., scheduled frames) by SRPT are undecodable at the receiver, because they rely on other more important frames that were deprived of channel time in the presence of small B (or P) frames. Therefore, there exists a significant gap between DFR and JFR for SRPT. This gap is the sign of channel time wastage by scheduling undecodable frames. F-SRPT also has the similar drawback, because it is essentially the same as SRPT (see Figure 3.5). It only maintains

fairness among high and low rate flows. Using the FDA technique, F-SRPT scheduler does not waste channel time by scheduling undecodable frames. Since these frames are mainly of the less important types, this would free up some channel time for scheduling frames of more important types. In effect, none of the undecodable frames are scheduled, and any scheduled frame is decodable. Therefore, JFR and DFR are identical when FDA technique is used.

As mentioned before, for small values of ϕ, the traffic load is very bursty and DFR is thus high. However, since PNC is in charge of admitting new flows to the system, it can alleviate this issue to some extent. For instance, it can reduce the burstiness of traffic by delaying the initiation of a flow for a few milliseconds, which is negligible from user's point of view. Therefore, we focus on the cases when ϕ is larger than superframe size, and present the performance evaluations averaged over 8 ms $< \phi <$ 30 ms. Figure 3.6 shows that dropping undecodable frames can reduce DFR of F-SRPT scheduler up to 61%. Note that FDA technique improves the objective measure of video quality, despite causing JFR to increase. As explained before, we are not concerned about an increase of JFR, as long as DFR decreases.

Using EDD along with SRPT, the EDD+SRPT scheduler gives higher priority to the frames with more stringent deadline. This approach reduces both JFR and DFR, compared to SRPT. The nature of SRPT, however, again causes a large gap between JFR and DFR. The FDA technique improves DFR of EDD+SRPT, basically in the same manner as described for SRPT. Figure 3.7 depicts the effect of ϕ and shows that

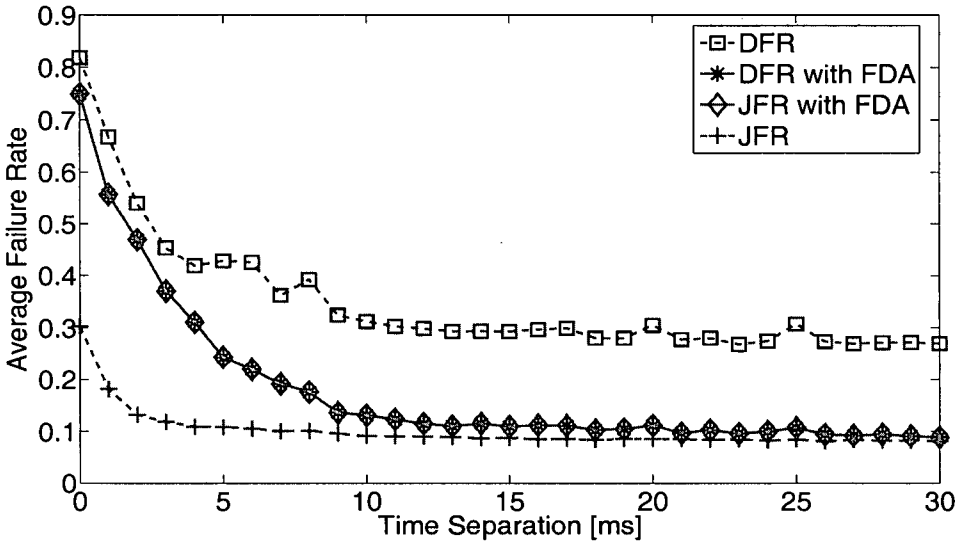

Figure 3.5: Effect of FDA technique on F-SRPT scheduling algorithm versus time separation ϕ.

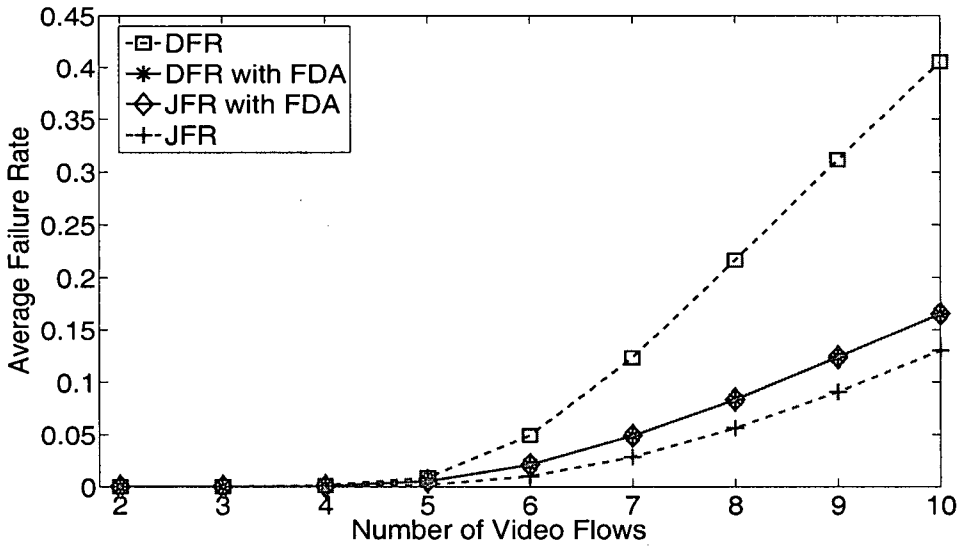

Figure 3.6: Effect of FDA technique on F-SRPT scheduling algorithm versus the number of flows F.

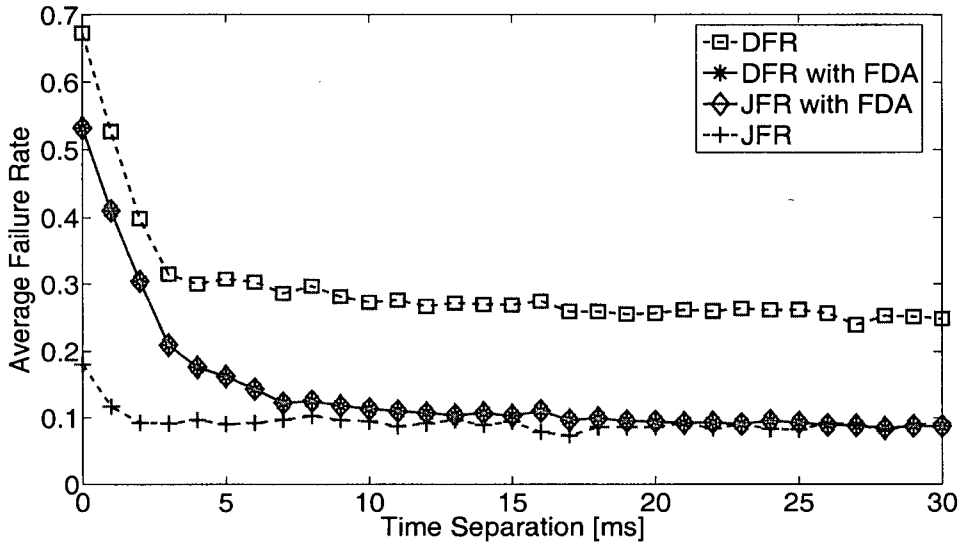

Figure 3.7: Effect of FDA technique on EDD+SRPT scheduling algorithm versus time separation ϕ.

FDA technique can significantly reduce DFR. Figure 3.8 depicts this improvement versus the number of video flows in the system, which is up 60% reduction in DFR.

Unlike the previous two schedulers, PAP scheduler treats video frame types differently, and uses the information about the type of frames. It serves I frames with the highest priority, followed by P and B frames, respectively. These priorities are based on the importance of each frame type. The FDA technique has the least impact on the PAP scheduler, and does not affect DFR for $\phi > 5$ ms (see Figure 3.9). Since PAP schedules I and P frames before B frames, most of the I and P frames are scheduled, and they have low rate of missing their deadline. As explained in Section 2.3.1, indirect loss is the result of expiry of I or P frames. Therefore, PAP scheduler results in a small number of undecodable frames, which reduces channel time wastage due to scheduling undecodable

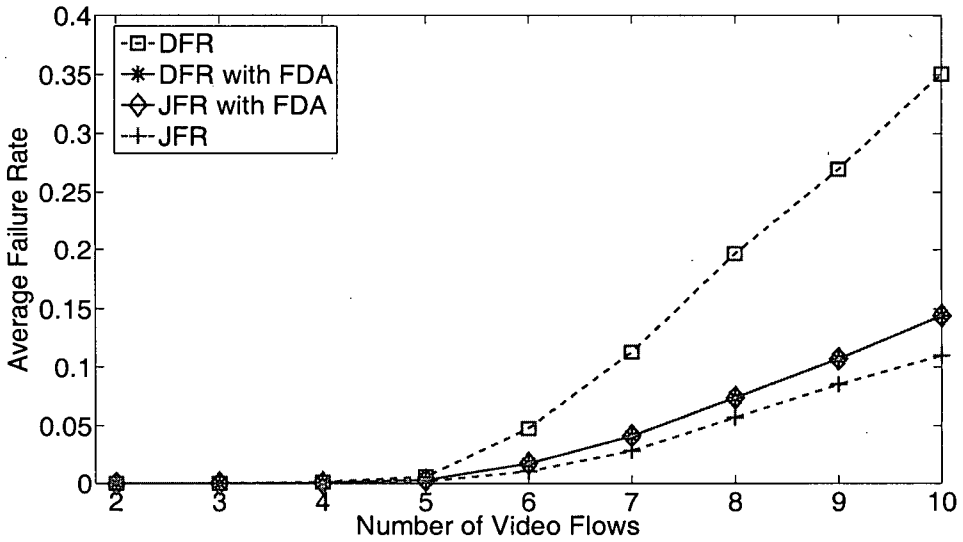

Figure 3.8: Effect of FDA technique on EDD+SRPT scheduling algorithm versus number of flows F.

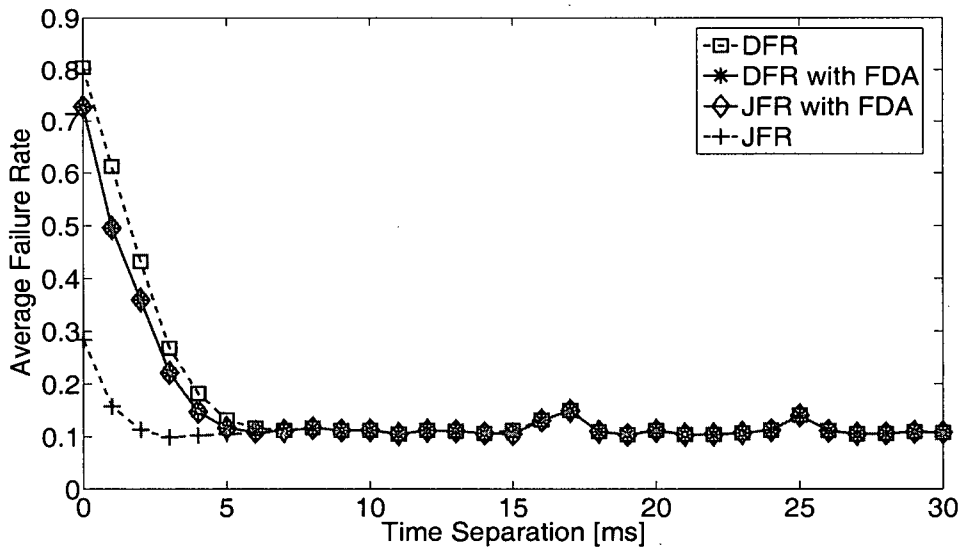

Figure 3.9: Effect of FDA technique on PAP scheduling algorithm versus time separation ϕ.

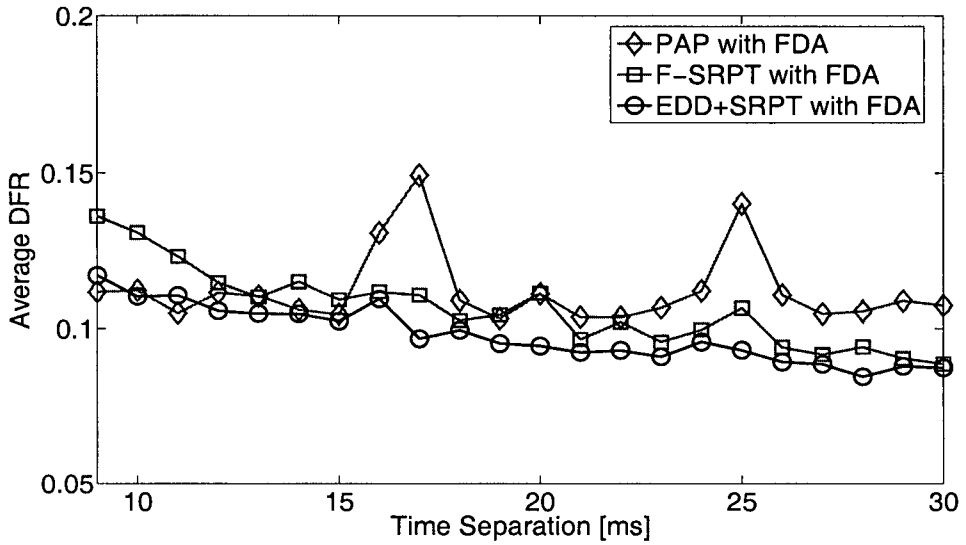

Figure 3.10: Comparison of F-SRPT, EDD+SRPT, and PAP schedulers when FDA is used.

frames. Thus, as shown in Figure 3.9, the gap between DFR and JFR for PAP scheduler is less than the other two schemes. So there is little room for improving PAP, as the FDA technique uses the wasted channel time to reduce DFR. Figure 3.10 illustrates the DFR of the three studied scheduling algorithms with FDA technique. It shows that none of the schedulers is the best for all values of ϕ.

3.4 Summary

In this chapter, we proposed an FDA technique to improve the performance of schedulers for video traffic. Our FDA technique can be applied to various types of schedulers and does not incur any extra signalling overhead between DEVs and the PNC, except for exchanging the information about frame type. We used both DFR and JFR as

performance metrics when evaluating video traffic schedulers, and emphasized that low DFR is a more appropriate indicator of the QoS given to video traffic. We analyzed the effect of FDA technique on F-SRPT, EDD+SRPT and PAP schedulers, and showed up to 61%, 60% reduction in DFR, for the first two schedulers.

Chapter 4

Scheduling Algorithm for Video Flows

4.1 Introduction and Motivations

In Chapter 3, we proposed the FDA technique for performance improvement. In addition, it was emphasized that DFR is a suitable performance metric for scheduling algorithms, and reflects the user-perceived QoS. We also presented the performance of three scheduling algorithms under the DFR metric. A natural question that may arise is that what is the best possible performance that we can obtain in terms of DFR metric. We shall address the answer to this question in this chapter.

The mathematical techniques that we used in this chapter were summarized in Sections 2.5 and 2.6. The rest of this chapter is organized as follows. In Section 4.2, we describe the MDP formulation for our scheduling problem [7]. Algorithm implementation and simulation results are presented in Sections 4.3 and 4.4, respectively. Finally, summary is given in Section 4.5.

4.2 Problem Formulation

In this section, we formulate the problem of finding the scheduling policy with minimum average DFR in the form of a Markov decision process (MDP) with average reward criterion [34]. We next relate the gain of the scheduling policy with the average DFR that it yields. As a result of the linear relationship between gain and average DFR, we can use reinforcement learning (RL) techniques [35][36] that find optimal gain policy, in order to find the scheduling policy which minimizes the average DFR.

4.2.1 Assumptions and System Model

We assume that the number of video flows, denoted by F, is fixed. For simplicity, it is assumed that the deadline of video frames is constant, and is equal to frame inter-arrival time γ. As a result, the queue of each flow holds no more than one frame at any time. We denote the GOP pattern of the i^{th} flow ($i \in \mathcal{F} \triangleq \{1, 2, \ldots, F\}$) by (N_i, M_i), where N_i and M_i are the I-to-I and I-to-P frame distance, respectively. The GOP pattern of all flows is assumed to remain constant during the flows' lifetime. We model the scheduling problem for video flows as an MDP. The scheduler is the decision making agent.

4.2.2 Decision Epochs

The scheduler should make a decision at the beginning of each superframe, and decide to schedule which flows in that superframe. The superframe size η is assumed to be fixed and is less than γ. For simplicity, we show the decision epochs in terms of superframe

size. For example, the actual time of decision epoch n, which is the beginning of n^{th} superframe, is $(n-1)\eta$. Therefore, the set of decision epochs is $\{1, 2, \ldots, T\}$.

4.2.3 States

Let the function $\text{tx}_i(x)$ give the amount of channel time that flow i requires to send x bits of data. The simplest form of this function is $\text{tx}_i(x) = x/\text{channel\_data\_rate}_i$. However, depending on the acknowledgement policy, inter-frame spacing times, and maximum MAC fragment size, this function can have a different form. Moreover, assume that the maximum frame size of flow i is L_i^{max}. We define the set of possible system states as follows:

$$\begin{aligned} \boldsymbol{S} \triangleq \Big\{ & \mathbf{s} = (\mathbf{l}, \mathbf{d}, \mathbf{g}, \boldsymbol{\delta}) \mid \forall\, i \in \mathcal{F}:\ 0 \le l_i \le \text{tx}_i(L_i^{max}), \\ & d_i \in \{0, \ldots, D^{max}\},\ g_i \in \{0, \ldots, N_i - 1\},\ \delta_i \in \{0, 1\} \Big\} \end{aligned} \tag{4.1}$$

where

- $\mathbf{l} \triangleq [l_1\ l_2\ \cdots\ l_F]$, and l_i is the channel time required by the frame in the queue of flow i, in milliseconds.
- $\mathbf{d} \triangleq [d_1\ d_2\ \cdots\ d_F]$, and d_i is the number of full superframes left until the arrival of the next frame. Since the maximum tolerable delay for video frames is the frame inter-arrival time, d_i can alternatively be interpreted as the deadline of the frame in the queue of flow i, in terms of superframe size.

- $\mathbf{g} \triangleq [g_1\ g_2\ \cdots\ g_F]$, and g_i is the offset of the frame in the queue of flow i, with respect to the beginning of the frame's GOP. For the first frame of the GOP $g_i = 0$, and for the last one $g_i = N_i - 1$.

- $\boldsymbol{\delta} \triangleq [\delta_1\ \delta_2\ \cdots\ \delta_F]$, and δ_i is set to 0 if any I or P frame, on which the current frame of flow i depends, is directly lost; and is equal to 1 otherwise (see Section 3.2).

- $D^{max} \triangleq \lfloor \frac{\tau}{\eta} \rfloor$, is the maximum frame deadline in units of superframe size η, where $\lfloor \cdot \rfloor$ is the floor function.

The scheduler incorporates the FDA technique, in order to determine the value of $\boldsymbol{\delta}$ based on $\mathbf{l}$, $\mathbf{d}$ and $\mathbf{g}$.

4.2.4 Actions

Let $\boldsymbol{A}$ be the set all possible actions:

$$\boldsymbol{A} = \left\{ \mathbf{a} \triangleq (a_1,\ a_2,\ \cdots\ a_F,\ p)\,|\,p \in \mathcal{F} \cap \{0\};\ a_i \in \{0,1\}, \forall i \in \mathcal{F};\ a_p = 0 \text{ if } p \neq 0 \right\}. \quad (4.2)$$

where a_i is equal to 1 if the scheduler allocates enough channel time to flow i so that it can *fully* transmit its frame. Otherwise, it is equal to 0. The parameter p is the flow that can only transmit parts of its frame during the channel time that the scheduler allocates to it. If no such flow exists, then $p = 0$. Here we assume that in each superframe, the scheduler allows at most one partial frame transmission, and the rest are full frame transmissions. The last conditional statement in equation (4.2) implies that the frame of flow p cannot be both fully and partially transmitted simultaneously.

At each decision epoch, i.e., the beginning of each superframe, the scheduler should choose an action depending on the current state. The chosen action must satisfy a few constraints. First, the channel time required to transmit scheduled frames should not be more than the superframe length η:

$$\sum_{i=1}^{F}(a_i \cdot l_i) \leq \eta. \tag{4.3}$$

Second, all the scheduled flows must be eligible:

$$a_i \leq e_i, \ \forall i \in \mathcal{F} \tag{4.4}$$

$$d_p > 1 \text{ and } e_p = 1 \quad \text{if } p \neq 0 \tag{4.5}$$

where e_i is the eligibility of the flow i, defined in Section 3.2. Equation (4.4) implies that $a_i = 0$, if $e_i = 0$. In other words, the ineligible flows are never scheduled. Equation (4.5) implies that p should be eligible and have a deadline greater than 1. Partial transmission of a frame with deadline one is wasting channel time, since the frame will expire at the end of the superframe. After the scheduler makes its decision about which frames should get fully transmitted in a superframe, there may still remain some channel time in that superframe that is not enough for full transmission of any unscheduled frame. This remaining time is given by:

$$l^{partial} = \eta - \sum_{i=1}^{F}(a_i \cdot l_i).$$

The fact that $l^{partial}$ is not enough for full transmission of any eligible frame that is not scheduled, can be formally expressed as the following constraint:

$$l_i > l^{partial}, \qquad \forall\, i \in \mathcal{F},\ e_i = 1,\ a_i = 0. \tag{4.6}$$

The idle channel time is allocated to the flow p. If no such flow exists, then p is set to 0.

Consequently, the set of possible actions in state $\mathbf{s} \in \boldsymbol{S}$, denoted by $\boldsymbol{A}_{\mathbf{s}}$, is the largest subset of $\boldsymbol{A}$, whose members satisfy all the constraints (4.3)–(4.6). These constraints guarantee that the scheduler accommodates as many eligible frames as possible. Note that $\boldsymbol{A}_{\mathbf{s}}$ is stationary and only depends on the system states.

4.2.5 Reward Function and Gain

As mentioned in Section 2.3.1, because of the hierarchical structure and interdependency of video frames, some frames may get indirectly lost. The reward function that we choose should account for this fact. We give a reward of one unit when a frame is scheduled. On the other hand, if the deadline of a frame expires, the scheduler receives a penalty (negative reward) of W units, where W is the number of frames that depend on the expired frame. Table 2.2 shows the number of dependencies between GOP frames of a (N, M) video flow.

Let $c_i(\mathbf{s})$ be the state-dependent penalty (or cost) function for flow i:

$$c_i(\mathbf{s}) = \begin{cases} \big(N_i + (M_i - 1)\big) \cdot e_i \cdot U_{\{d_i=1\}}, & \text{if } y_i(g_i) = I, \\ e_i \cdot U_{\{d_i=1\}}, & \text{if } y_i(g_i) = B, \\ \big(N_i - 1 - (\frac{g_i}{M_i} - 1)M_i\big) \cdot e_i \cdot U_{\{d_i=1\}}, & \text{if } y_i(g_i) = P, \end{cases}$$

where the function $y_i(g) : \{0, \ldots, N_i - 1\} \rightarrow \{I, B, P\}$ maps g to frame types as follows:

$$y_i(g) = \begin{cases} I, & \text{if } g = 0, \\ B, & \text{if } g \bmod M_i \geq 1, \\ P, & \text{if } g \bmod M_i = 0 \text{ and } g \neq 0. \end{cases}$$

Furthermore, $U_{\{.\}}$ is the indicator function and is equal to 1 if its argument is true, and is 0 otherwise. The product $e_i(n) \times U_{\{d_i(n)=1\}}$ indicates if the flow i has an urgent eligible frame. Hence, the scheduler should receive $c_i(\mathbf{s}(n))$ units of penalty if it does not schedule the frame of flow i in the current superframe.

As a result, we can express the state- and action-dependent reward that the scheduler receives at superframe n by:

$$r(\mathbf{s}(n), \mathbf{a}(n)) = \sum_{i=1}^{F} \Big[a_i(n) - c_i(\mathbf{s}(n))(1 - a_i(n)) \Big] \tag{4.7}$$

In order to show the merits of the reward function in equation (4.7), we study the policy gain that it yields. The gain of policy π under the average reward criteria is the average accumulated reward. In our formulation, the policy gain is given by:

$$\begin{aligned} \rho^{\pi} &= \frac{1}{T} \sum_{n=1}^{T} r(\mathbf{s}(n), \mathbf{a}(n)) \\ &= \sum_{i=1}^{F} \left(\frac{\sum_{n=1}^{T} a_i(n) - \sum_{n=1}^{T} c_i(\mathbf{s}(n))(1 - a_i(n))}{T} \right) \end{aligned} \tag{4.8}$$

The total number of frames for each flow is $\text{total\_frames} = \frac{\text{total\_time}}{\text{inter\_arrival\_time}} = \frac{\eta T}{\gamma}$. Furthermore, the terms $\sum_{n=1}^{T} a_i(n)$ and $\sum_{n=1}^{T} c_i(\mathbf{s}(n))(1 - a_i(n))$ in equation (4.8) are in fact the total number of scheduled frames and the total number of undecodable frames for each

flow, respectively. In addition, these two terms sum up to the total number of frames for flow i. Consequently, equation (4.8) can be rewritten as:

$$\begin{aligned}
\rho^{\pi} &= \frac{\eta}{\gamma}\sum_{i=1}^{F}\Big(\frac{\text{total_scheduled} - \text{total_undecodable}}{\text{total_frames}}\Big) \\
&= \frac{\eta}{\gamma}\sum_{i=1}^{F}\Big(1 - \frac{2 \times \text{total_undecodable}}{\text{total_frames}}\Big) \\
&= \frac{\eta}{\gamma}\sum_{i=1}^{F}(1 - 2\text{DFR}_i^{\pi}) \\
&= \frac{\eta F}{\gamma} - \frac{2\eta}{\gamma}\sum_{i=1}^{F}\text{DFR}_i^{\pi} \qquad (4.9)
\end{aligned}$$

where DFR_i^{π} is the decoding failure rate of flow i under the policy π. Let $\overline{\text{DFR}}^{\pi} \triangleq \frac{1}{F}\sum_{i=1}^{F}\text{DFR}_i^{\pi}$ denote the average DFR under the policy π. Using equation (4.9), we have

$$\overline{\text{DFR}}^{\pi} = \frac{1}{2} - \frac{\gamma}{2\eta F}\rho^{\pi}. \qquad (4.10)$$

Equation (4.10) shows that in our formulation, the average DFR is a linear function of gain. Therefore, we conclude that:

$$\arg\max_{\pi}\rho^{\pi} = \arg\min_{\pi}\overline{\text{DFR}}^{\pi}. \qquad (4.11)$$

Hence, an optimal (maximum) gain policy yields the optimal (minimum) average DFR, which is what we aim to find.

4.2.6 State Transitions

Assume that at superframe n, the system is in state $\mathbf{s}(n)$ and chooses the action $\mathbf{a}(n)$. In this section, we determine the system state at superframe $n+1$, $\mathbf{s}(n+1)$. This state

depends on $\mathbf{s}(n)$, $\mathbf{a}(n)$ and new frame arrivals. We determine the state transitions per flow, i.e., we show how the state variables related to each flow change. The whole system state is updated by performing the same procedure for all the flows:

When $d_i(n) \geq 1$, no new video frame will arrive for flow i. So, the remaining time until the next arrival is reduced by one superframe. If the flow is scheduled, its length will become 0. And if it is partially transmitted, its length will reduce by $l^{partial}$. Otherwise, the length remains unchanged. The frame offset within GOP also does not change. The value of δ_i changes only when an I or P frame of flow i expires, or when a new GOP starts, which is indicated by arrival of a new I frame for flow i. None of these happens when $d_i(n) \geq 1$. Hence, if $d_i(n) \geq 1$, then the state *deterministically* changes as follows:

$$\begin{aligned} d_i(n+1) &= d_i(n) - 1 \\ l_i(n+1) &= l_i(n)(1 - a_i(n)) - l^{partial} U_{\{i=p\}} \\ g_i(n+1) &= g_i(n) \\ \delta_i(n+1) &= \delta_i(n). \end{aligned}$$

When $d_i(n) = 0$, it means that the frame in the queue of flow i has expired, if it has not already been sent, i.e., if $l_i(n) \neq 0$. It also means that a new frame will arrive for flow i, within superframe n. We denote the deadline and channel time requirement of the newly arrived frame for flow i within superframe n by $d_i^{new}(n)$ and $l_i^{new}(n)$, respectively.

Thus, when $d_i(n) = 0$ the state changes with probability $\text{prob}_i(l_i^{new}(n))$ as follows:

$$
\begin{aligned}
d_i(n+1) &= d_i^{new}(n) \\
l_i(n+1) &= l_i^{new}(n) \\
g_i(n+1) &= (g_i(n)+1) \bmod N_i \\
\delta_i(n+1) &= \begin{cases} 1, & \text{if } y_i(g_i(n+1)) = I, \\ 0, & \text{if } y_i(g_i(n)) = I, P \text{ and } l_i(n) \neq 0, \\ \delta_i(n), & \text{otherwise,} \end{cases}
\end{aligned}
$$

where $\text{prob}_i(l_i^{new}(n))$ is the probability of the channel time request of flow i be $l_i^{new}(n)$. This probability is simply related to the frame size distribution probability of flow i.

4.3 Algorithm Implementation

In this section, we discuss how to find the optimal gain policy defined by the MDP given in the previous section. Our formulation has modeling and dimensionality issues, so DP cannot be used. First, we do not have the knowledge of the state transition probabilities, because the scheduler does not know the frame size distribution of the video flows beforehand. We use the SMART algorithm [38] to overcome the curse of modeling. Second, in the state space defined in equation (4.1) we have one continuous variable, i.e. the frame length. Even if we quantize this variable to several levels, the state space would still be huge. For instance, suppose that $D^{max} = 4$ and $N_i = 12$ for all $i \in \mathcal{F}$, and we quantize l_i's to 8 levels. Thus, $|\boldsymbol{S}| = (8 \times 5 \times 12 \times 2)^F \approx 1000^F$. Furthermore, it is easy to verify that $|\boldsymbol{A}| = F \cdot 2^F$. According to equation (2.4), SMART

learns the Q-function Q: $\boldsymbol{S} \times \boldsymbol{A} \rightarrow \mathbb{R}$, which has impractical memory requirement and convergence rate given the size its domain. We solve the problem in two phases. In Section 4.3.1, we describe a compact representation of the state-action space, which is a mapping from $\boldsymbol{S} \times \boldsymbol{A}$ to a smaller set $\boldsymbol{\Omega}$. Then, in Section 4.3.2, a memory-based function approximator is used to map $\boldsymbol{\Omega}$ to the set of real numbers $\mathbb{R}$.

4.3.1 Features and State Space Representation

We use an aggregation scheme [36], which gives a more compact representation of the state-action space. This scheme is useful when we have knowledge of an initial part of the environment's dynamics but not necessarily of the full dynamics. For instance, in our formulation, the effect of chosen action on the system state can be *deterministically* simulated by the scheduler. On the other hand, the new frame arrivals cannot be deterministically simulated. Using this scheme, we obtain a more efficient learning method by breaking the environment's dynamics into the immediate effect of the action, which is deterministic and perfectly known, and the arrival process of new frames with unknown size distribution [36]. We refer to the system state *after* the effect of the taken action as *afterstate* [36]. The actual system state in the next superframe is the afterstate updated according to the new frame arrivals.

In order to explain how afterstates can cause aggregation within the state-action space, consider the case when at superframe n the scheduler can accommodate all the eligible frames in the system (i.e., when the system is under-loaded). Examples of this

case are:

- Flow 1 has a frame with $l_1 = 2$, $d_1 = 2$, $g_1 = 0$, and $\delta_1 = 1$. The rest of the flows have no frame and their queues are empty;
- Flow 2 has a frame with $l_2 = 1$, $d_2 = 3$, $g_2 = 5$, and $\delta_2 = 1$. The rest of the flows have no frame and their queues are empty;
- Flows 1 and 3 have a frame with $l_1 = 0.5$, $d_1 = 2$, $g_1 = 3$, $\delta_1 = 1$, $l_3 = 2.5$, $d_3 = 1$, $g_3 = 7$, and $\delta_3 = 1$. The rest of the flows have no frame and their queues are empty.

In any of the above examples, the scheduler's action is that all the eligible frames be transmitted in superframe n. The effect of this action is that the system will become empty. Thus, the afterstate (i.e. the system state after the effect of the action) corresponds to an empty system. As a result, all of the state-action pairs, which correspond to an under-loaded system becoming empty, have the same afterstate and can be aggregated. Similarly, any afterstate represents all the state-action pairs that result in that afterstate. Using afterstates is in fact a mapping from $\boldsymbol{S} \times \boldsymbol{A}$ to $\boldsymbol{S}$. Reference [51] successfully applies this idea to admission control problem in cellular networks.

The state space of our MDP has binary variables in $\boldsymbol{\delta}$, continuous variables in $\mathbf{l}$, and integer variables in $\mathbf{d}$ and $\mathbf{g}$. Even after using afterstates, the scheduler should learn the Q-function over $\boldsymbol{S}$, which is still very large. Therefore, we need to specify the important features in each afterstate, which most affect the scheduling decision. As mentioned before, there is no systematic way of choosing the features. We approach this task by

studying the behavior of the SRPT, EDD and PAP schedulers. All of these schedulers can be viewed as single-stage decision making process, where the scheduler chooses an action merely based on its immediate effect, i.e., based on the resulting afterstate. SRPT uses information about length. Since it minimizes the number of pending frames in the system, it chooses an action that yield an afterstate with minimum number of frames. In other words, the only feature of the afterstate that is important for SRPT is the number of frames in the system. Similarly, EDD+SRPT uses the number of frames in the system, but groups them according to their deadlines. It chooses an action that leads to the afterstate with the minimum number of frames in the group with the smallest deadline. If this value is the same for the afterstates of two actions or more, the size of the group with second smallest deadline is used, and so on. Therefore, the features that EDD+SRPT used are the number of frames in each deadline group. PAP, which is an application-aware scheduler, further partitions the frames in the system according to their type. In other words, SRPT, EDD+SRPT, and PAP schedulers mainly focus on the information about length, deadline, and type, respectively. With the addition of FDA technique, they can take advantage of decodability as well. Since the optimal scheduler should focus on all the information, it is reasonable to assume that it uses all the features used by those schedulers. In light of these similarities, the following features are chosen:

- $F_{y,\,non\_urgent} = \left|\{i \mid l_i > 0,\ y_i(g_i) = y,\ \delta_i = 1,\ d_i > 1\}\right|$ is the number of all the frames of type y in the system that have a deadline greater that 1.

- $F_{y,\,urgent} = \left|\left\{i \mid l_i > 0,\ y_i(g_i) = y,\ \delta_i = 1,\ d_i = 1\right\}\right|$ is the number of all the frames of type y in the system that have a deadline equal to 1.

- $F_{y,\,lost} = \left|\left\{i \mid l_i > 0,\ y_i(g_i) = y,\ d_i = 0\right\} \cup \left\{i \mid l_i > 0,\ y_i(g_i) = y,\ \delta_i = 0\right\}\right|$ is the number of all the frames of type y in the system that are either expired or indirectly lost.

The parameter y can be substituted with I, P, and B. Thus, in total we represent each afterstate using 9 (i.e., 3×3) features. A vector whose elements are the features is referred to as a feature vector. Since in our formulation every feature can vary from 0 to F, the set of all feature vectors is $\mathbf{\Omega} = \{0, 1, \ldots, F\}^9$.

So far in this section, we have performed two aggregation steps. The first one is the use of afterstates, which is a function from $\boldsymbol{S} \times \boldsymbol{A}$ to $\boldsymbol{S}$. The second aggregation is feature extraction, which is a function from $\boldsymbol{S}$ to $\mathbf{\Omega}$. Using the procedures described in this subsection, we can define the function f: $\boldsymbol{S} \times \boldsymbol{A} \rightarrow \mathbf{\Omega}$, where $\mathbf{x} = f(\mathbf{s}, \mathbf{a})$ is the feature vector of the afterstate resulted by taking action $\mathbf{a}$ in state $\mathbf{s}$. In the rest of this section, we describe how we can perform generalization (i.e., function approximation) over the set of feature vectors $\mathbf{\Omega}$.

4.3.2 Function Approximation: Kanerva Coding

As the dimension of the state space grows, the complexity of many function approximators, such as tile coding, increases exponentially with it. However, the complexity of the target function can be unrelated to the size and dimensionality of the state space, and

need not grow exponentially with it [36]. For instance, we might have a 100-dimensional state space where only one of the dimensions happens to affect the target function. Kanerva coding is a memory-based function approximator that allows us to deal with the complexity of a reasonable approximation of the target function, and not the dimensionality of the state space [36]. It was originally presented in the context of sparse distributed memory [52]. Like other memory-based techniques, it has nice memory requirement and generalization properties, and also allows dynamic memory allocation. The strength of Kanerva coding increases with the number of its memory locations (i.e., prototypes), which can be set independent of the dimensionality of the state space. The more the number of prototypes, the more the complexity of the target function that it can approximate. Kanerva coding is successfully applied in the context of RL in [53] and [54].

In our scheduling problem, we use Kanerva coding to approximate the function $\hat{Q}: \Omega \rightarrow \mathbb{R}$, which gives the value of a feature vector. The prototypes of Kanerva coding are feature vectors in Ω. The value of any feature vector in Ω is then approximated using the values of those prototypes. In order to do so, a similarity metric should be defined. We define the distance of two feature vectors $\mathbf{x}, \mathbf{z} \in \Omega$ as the number of unequal features in the two vectors, i.e., $\text{dist}(\mathbf{x}, \mathbf{z}) \triangleq \sum_{j=1}^{|\mathbf{x}|} U_{\{x_j \neq z_j\}}$ [54]. Furthermore, we define the similarity of two feature vectors $\mathbf{x}$ and $\mathbf{z}$ as follows [55]:

$$\lambda(\mathbf{x}, \mathbf{z}) = \begin{cases} 1 - \dfrac{\text{dist}(\mathbf{x}, \mathbf{z})}{\beta}, & \text{dist}(\mathbf{x}, \mathbf{z}) \leq \beta \\ 0, & \text{otherwise} \end{cases}$$

where β is the activation radius. In other words, two feature vectors have no similarity if they have more than β different features. Let $\mathbf{h}_k$ be the kth prototype in Kanerva coding. To calculate the value of a feature vector $\mathbf{x}$, we first determine the set of prototypes that are activated by $\mathbf{x}$, which are similar to it. We denote this set by $H_{\mathbf{x}}$ and formally define it as $H_{\mathbf{x}} = \{k \mid \lambda_k > 0\}$, where $\lambda_k \triangleq \lambda(\mathbf{x}, \mathbf{h}_k)$ is the similarity between $\mathbf{x}$ and $\mathbf{h}_k$. Let w_k be the value corresponding to $\mathbf{h}_k$. Then the value of $\mathbf{x}$ is given by

$$\hat{Q}(\mathbf{x}) = \frac{\sum_{k \in H_{\mathbf{x}}} \lambda_k w_k}{\sum_{k \in H_{\mathbf{x}}} \lambda_k}. \tag{4.12}$$

Thus, the function $\hat{Q}(\cdot)$ is a linear combination of the value of prototypes. We now give the update procedure that should be used in place of equation (2.4). Using the standard gradient descent algorithm [36] for linear approximation, the update equation becomes [55]:

$$w_i \leftarrow w_i + \mu \frac{\lambda_i}{\sum_{k \in H_{\mathbf{x}}} \lambda_k} \left\{ \left[r(\mathbf{s}, \mathbf{a}) - \eta\rho + \max_{\mathbf{a}'} \hat{Q}(f(\mathbf{s}', \mathbf{a}')) \right] - \hat{Q}(f(\mathbf{s}, \mathbf{a})) \right\}, \forall i \in H_{\mathbf{x}}. \tag{4.13}$$

Since in our formulation, the immediate reward does not depend on $\mathbf{s}'$, we have used $r(\mathbf{s}, \mathbf{a})$ instead of $r(\mathbf{s}, \mathbf{a}, \mathbf{s}')$.

The distribution of prototypes in the state space is crucial for the approximation accuracy. In our problem, it is not efficient to cover the whole state space, because the scheduler does not necessarily explore all of it. Instead, we use a dynamic prototype addition method, which covers the vicinity of the trajectory that the scheduler explores within the state space. We follow the approach given in [55]. The dynamic addition algorithm starts with an empty set of prototypes, and builds up over time. If the set is too

sparse around an encountered feature vector, enough prototypes are added. Furthermore, the prototypes should not be too close (similar) to each other. Therefore, the following condition is imposed on the similarity of prototypes:

$$\lambda(\mathbf{h}_i, \mathbf{h}_j) \leq \begin{cases} 1 - \frac{1}{K-1}, & K \geq 3 \\ 0.5, & K = 2 \end{cases} \tag{4.14}$$

for any pair of memory locations $\mathbf{h}_i$ and $\mathbf{h}_j$. K is the minimum number of prototypes that we aim to be activated for a feature vector. In other words, we want to have at least K prototypes in a sphere of radius β centered at any encountered feature vector. As a result, K and β determine the minimum density of prototypes along the scheduler's trajectory in the state space. The more the density, the more prototypes we have, and the more accurate the approximation will be. The dynamic prototype addition algorithm has the following rules:

Rule 1: If fewer than K prototypes are activated by the input feature vector $\mathbf{x}$, add $\mathbf{x}$ to the set of prototypes, if its addition does not violate the condition (4.14).

Rule 2: If after applying Rule 1, the number of active prototypes (i.e., $|H_{\mathbf{x}}|$) is $K_0 < K$, then $(K - K_0)$ prototypes are randomly added within the neighborhood of $\mathbf{x}$. The neighborhood of $\mathbf{x}$ is the set of all feature vectors that are similar to it, i.e., $\{\mathbf{z} \mid \lambda(\mathbf{x}, \mathbf{z}) > 0\}$. Condition (4.14) is enforced when adding new prototypes. The value of a newly added prototype is set to the value of the function $\hat{Q}$ evaluated at that prototype according to equation (4.12).

For discussion on how random selection of prototypes can give good approximation,

see [56]. Successful application of this algorithm to RL tasks can be found in [53].

4.3.3 RL Scheduler

Since we use average reward optimality criterion, SMART [38] is a suitable model-free learning algorithm to find the optimal gain policy. Using SMART, we determine the scheduling policy that achieves the minimum average DFR. We call this scheduler as the *RL scheduler*. The pseudo-code of the RL scheduler is given in Figure 4.1.

At superframe $n = 0$, i.e., when the RL scheduler begins operating, cumulative reward CR and average reward ρ are both set to 0. The cumulative time CT is the duration over which CR is accumulated. We approximate the Q-value of actions with Kanerva coding. The set of prototypes of Kanerva coding is also empty at the beginning. After initialization, the RL scheduler makes scheduling decisions at the beginning of each superframe based on the following procedure. It first determines the exploration probability and the learning rate based on Darken-Chang-Moody (DCM) search-then-converge procedure [57]. Using DCM method, the exploration probability and learning rate at superframe n, i.e., q_n and μ_n, are given by $q_0/[1+(\frac{n^2}{q_r+n})]$ and $\mu_0/[1+(\frac{n^2}{\mu_r+n})]$, respectively. The parameters q_0, q_r, μ_0, and μ_r are constants. Afterwards, with probability $1-q_n$, the RL scheduler chooses the greedy action; with probability q_n, a random exploratory action (i.e., any action other than the greedy one) is chosen. In Step 3, the RL scheduler executes the chosen action and calculates the immediate reward (see equation (4.7)) and the next system state $\mathbf{s}'$. In step 4, the scheduler updates the set of prototypes according

Initialize the superframe number $n = 0$, cumulative reward $CR = 0$, cumulative time $CT = 0$, and the average reward $\rho = 0$. In addition, the set of prototypes is initialized as the empty set. The Superframe size is η. Suppose that the system starts in state $\mathbf{s}$.

while $n <$ MAX_STEPS do

1. Calculate exploration probability q_n and learning rate μ_n using the DCM method.
2. With probability $1 - q_n$, choose the greedy action $\mathbf{a} \in \boldsymbol{A}_{\mathbf{s}}$ that maximizes $\hat{Q}(f(\mathbf{s}, \mathbf{a}))$; otherwise, choose a random exploratory action from the set $\{\boldsymbol{A}_{\mathbf{s}} \setminus \mathbf{a}\}$.
3. Execute the chosen action. Let the system state at the next superframe be $\mathbf{s}'$, and the immediate received reward be $r(\mathbf{s}, \mathbf{a})$.
4. Apply Rule 1 and Rule 2 of Section 4.3.2 to the feature vector $\mathbf{x} = f(\mathbf{s}, \mathbf{a})$.
5. Update the weight of the prototypes according to:

$$w_i \leftarrow w_i + \mu_n \frac{\lambda_i}{\sum_{k \in H_{\mathbf{x}}} \lambda_k} \left\{ \left[r(\mathbf{s}, \mathbf{a}) - \eta\rho + \max_{\mathbf{a}'} \hat{Q}(f(\mathbf{s}', \mathbf{a}')) \right] - \hat{Q}(f(\mathbf{s}, \mathbf{a})) \right\}, \forall i \in H_{\mathbf{x}}$$

6. **if** a greedy action was chosen in Step 2, **then**
   Update $CT \leftarrow CT + \eta$;
   Update $CR \leftarrow CR + r(\mathbf{s}, \mathbf{a})$ and $\rho \leftarrow CR/CT$;
   else go to Step (2).
7. Go to the next superframe, i.e., update $n \leftarrow n + 1$ and $\mathbf{s} \leftarrow \mathbf{s}'$.

Figure 4.1: Pseudo-code of the RL scheduler.

to Rule 1 and Rule 2 of Section 4.3.2. The weights of the prototypes are updated in Step 5. This update is the same as equation (4.13). In step 6, the cumulative reward and average reward are updated only if a greedy action was chosen. Finally, in step 7, the scheduler goes to the next superframe and repeat the same procedure again. Figure 4.2 illustrates the structure of our proposed RL scheduler.

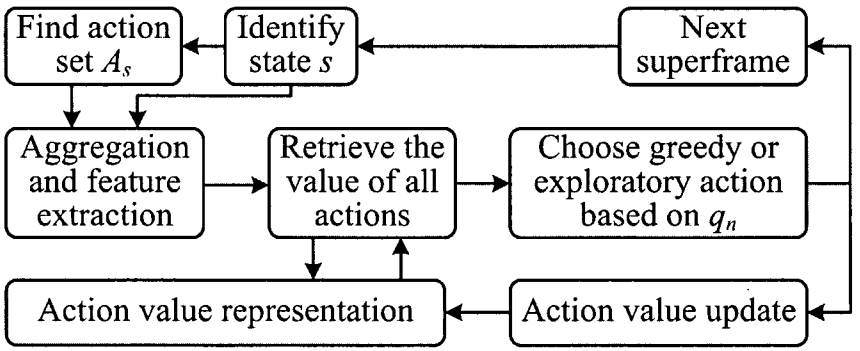

Figure 4.2: Structure of the RL scheduler.

4.4 Simulation Results

In Section 4.2, we presented an MDP formulation for finding the scheduling policy with minimum average DFR. In Section 4.3, we described a few simplification and approximation steps to make the problem solvable by RL techniques in practical time. As a result, we cannot make any rigorous claim on the optimality of the final solution given by the SMART algorithm. Nevertheless, we show through simulation that the final policy, hereafter referred to as the RL scheduler, has a significantly lower DFR than F-SRPT, EDD+SRPT, and PAP schedulers. In addition, we show that the RL scheduler is near-optimal in one special case. The simulator was implemented using MATLAB.

In our simulations, the first iteration (or simulation run) corresponds to a 500 s long scenario with F video flows and with start time separation ϕ. In the subsequent iterations, the scheduler performs in the same scenario, but this time it can do a better job based on its experiences from the previous iterations. With superframe size $\eta =$ 8 ms, each iteration would consist of $\frac{500}{8\times10^{-3}} =$ 62,500 superframes. The parameters

q_0 and μ_0 of the SMART algorithm, depend on how fast the scheduler can learn the optimal policy. The more the value of these parameters, the longer the scheduler should search and explore to find the optimal policy. In simulations, we set $q_0 = \mu_0 = 0.1$ and $q_r = \mu_r = 10^{10}$, as it gives the RL scheduler enough time to explore and find the optimal policy without too many iterations. We end the simulation when the learning rate μ_n and the exploration probability q_n fall below 0.005. Hence, there is no need to explicitly set MAX_STEPS in Figure 4.1. Like the parameters of DCM, the parameters of dynamic prototype addition for Kanerva coding can be found with experimentation. We found that $\beta = 3$ and $K = 8$ provides a sufficient prototype density for a reasonable approximation of the value-function corresponding to the feature vectors. The rest of the simulation parameters are the same as in Chapter 3 and are summarized in Table 3.1.

As mentioned in the previous chapter, the start time separation of the flows in the system can greatly affect the average DFR. The special case of interest is when $\phi = 0$, i.e., when all the flows in the system start at the same time. As a result, at the beginning of every superframe, all the frames in the system have the same deadline and type. Therefore, the deadline and type information is of no use for the scheduler. The scheduler has as much as one inter-arrival time to schedule the frames that arrive together. Obviously, the optimal scheduler in this case must leave the least possible number of frames behind at the end of this period. As mentioned earlier in Section 2.4.2, SRPT scheduler has this property, and is the optimal scheduler for $\phi = 0$. Consequently, we know exactly the optimal policy for $\phi = 0$, and can compare our RL scheduler

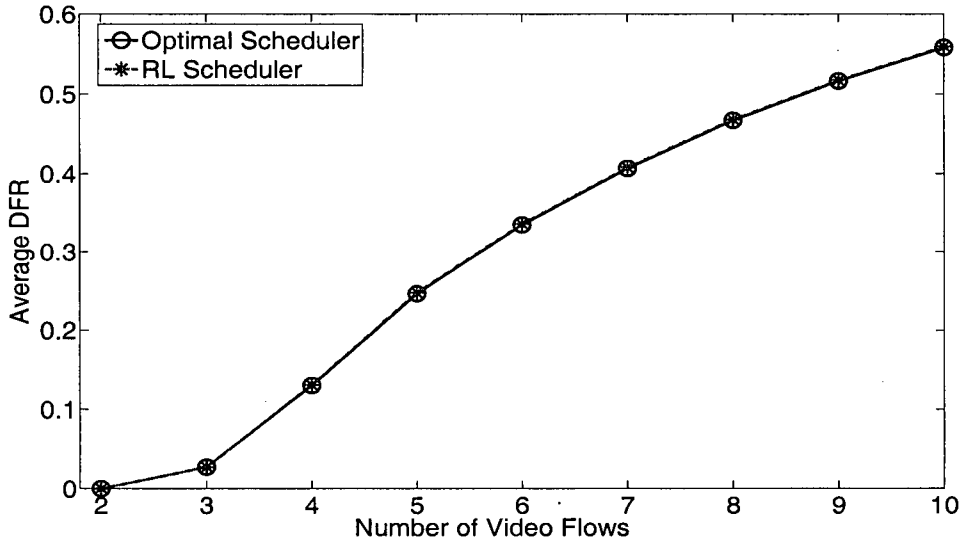

Figure 4.3: Comparison of RL scheduler and optimal scheduler for $\phi = 0$. RL scheduler is nearly optimal in this case.

with the optimal scheduler. Figure 4.3 shows that RL scheduler is nearly optimal for all the simulated values of F and achieves the minimum DFR. For $\phi = 0$, we do not expect any improvement beyond the performance of the optimal scheduler. However, for other values of ϕ, we cannot pursue the same approach, as the optimal scheduling policy is not known. Instead, we compare the performance of the RL schedulers with the other three schedulers (F-SRPT, EDD+SRPT, PAP). Similar to the approach in Chapter 3, the subsequent results given for average DFR are the mean of the results for 8 ms $< \phi \leq$ 30 ms.

Figure 4.4 shows this effect on F-SRPT, EDD+SRPT, and PAP schedulers with FDA. As one can see, for some values of ϕ PAP has the smallest average DFR, while for some other values EDD+SRPT has the smallest average DFR. On the other hand, the RL

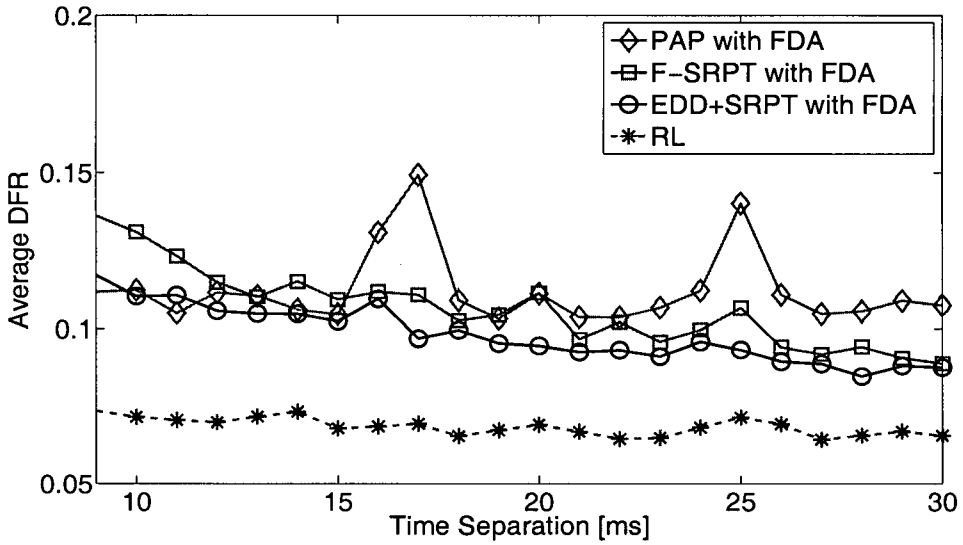

Figure 4.4: Effect of time separation on average DFR for $F = 9$. The RL scheduler has the smallest average DFR for $\phi > 8$ ms.

scheduler performs better than the other three for all values of ϕ, and is less sensitive to ϕ.

Figure 4.5 illustrates the average DFR for all four schedulers, including the RL scheduler. Similar to previous results, the RL scheduler has the smallest average DFR for all values of F. It has up to 42%, 49%, and 53% less average DFR when compared to EDD+SRPT, PAP, and F-SRPT schedulers, respectively.

Table 4.4 shows that reduction of the average DFR can also be translated to system capacity enhancement. Suppose that the acceptable user perceived quality is equivalent to the average DFR being less than 5%. Thus, the capacity of the system can be defined as the number of video flows that can be admitted to the system, while the average DFR is less than the maximum allowable value of 5%. Using this definition, the system capacity

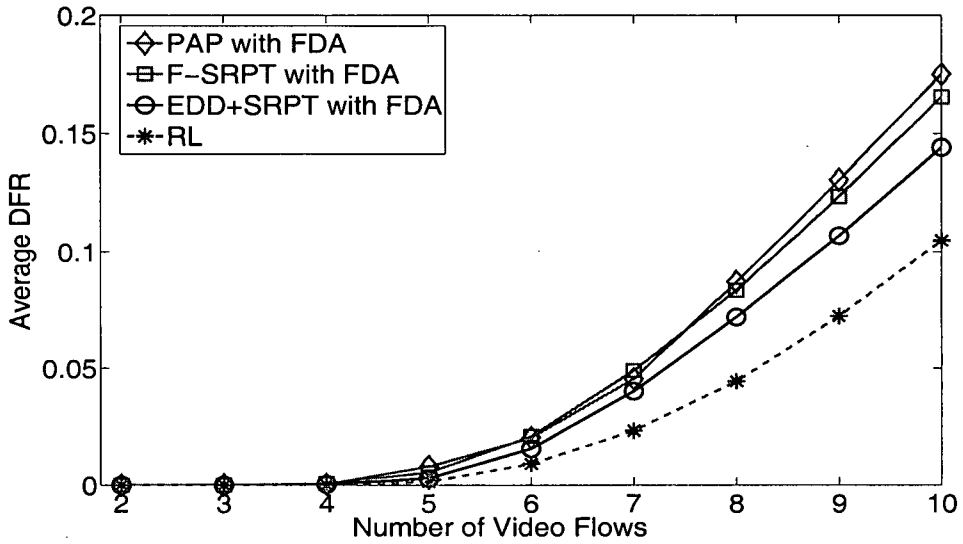

Figure 4.5: Comparison of RL scheduler and other schedulers.

is 7 flows for the conventional schedulers, as opposed to 8 flows for the RL scheduler. Consequently, in this example, the RL scheduler increases the system capacity by 14.3%.

As mentioned in Section 4.2.5, the policy gain and DFR have a linear relationship. We can verify the validity of equation (4.10) as follows. First, the estimated average

Number of Flows F	6	7	8	9	10
Minimum Average DFR (%)	1.5	4.0	7.2	10.7	14.4
RL Average DFR (%)	0.9	2.3	4.4	7.2	10.5
Absolute Reduction (%)	0.6	1.7	2.8	3.4	3.9
Relative Reduction (%)	42	42	38	32	27

Table 4.1: Quantitative comparison among RL scheduler and conventional schedulers. The minimum average DFR given by F-SRPT, EDD+SRPT, and PAP is used for comparison.

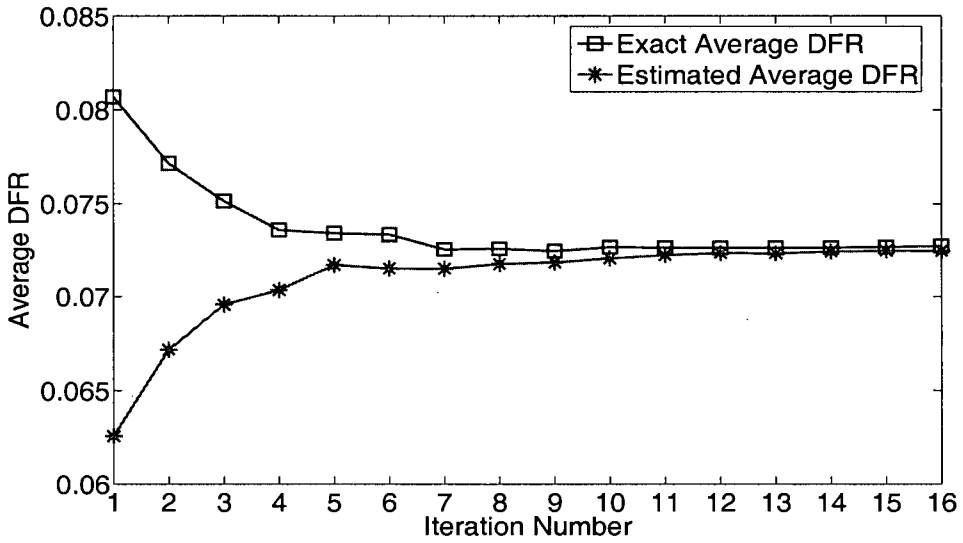

Figure 4.6: Comparison of the exact and the estimated average DFR ($F = 9$).

DFR is calculated by substituting the policy gain ρ in equation (4.10). RL schedulers calculates the value of ρ as part of its algorithm. Second, the exact average DFR is measured by counting the number of scheduled frames. Figure 4.6 compares these two values. As one can see, $\overline{\text{DFR}}$ in equation (4.10) under-estimates the exact average DFR, because the gain is only updated when the scheduler takes a greedy action. However, both greedy and exploratory actions affect the exact average DFR. Over time, with more iterations, as the RL scheduler learns the optimal policy and the exploration probability decays, the exact and estimated average DFR converge together. This result verifies the fact that the optimal gain policy yields the minimum average DFR.

4.5 Summary

In this chapter, we presented an MDP framework for finding the optimal scheduling algorithm with respect to average DFR. However, due to curse of modeling and curse of dimensionality, we appealed toward RL techniques. A model-free RL algorithm, called SMART, was used to overcome the modeling issue. We further used afterstates and features to reduce the dimensions of the state space and represent it in a more compact and suitable form. Moreover, Kanerva coding along with a dynamic prototype addition algorithm was used to the approximate the value-function of feature vectors. The final policy, i.e., the RL scheduler, yields a lower DFR and higher system capacity compared to its conventional counterparts.

Chapter 5

Conclusions and Future Work

In this thesis, we examined the problem of scheduling video flows in high-rate WPANs. Our aim was to improve QoS measured by low DFR, since we have shown that DFR is more appropriate than JFR. Our contributions in this thesis can be summarized as follows:

- We proposed a simple and efficient FDA technique for using the information about frame type at the MAC layer. Using the FDA technique, the scheduler can determine whether the receiver of a video frame has all the required information to decode that frame. This technique minimizes the channel time wastage caused by scheduling undecodable frames, and consequently, improves the QoS.

- The FDA technique is independent of the scheduler and can be used along with different scheduling algorithms. Simulation results showed that when FDA technique is applied to the F-SRPT [1] and EDD+SRPT [2] schedulers, there are up to 61% and 60% reduction in DFR, respectively.

- We formulated the scheduling of video flows in high-rate WPANs as an MDP problem. This model incorporates the decodability information extracted by the FDA

technique. It also takes into account the number and pattern of video flows, and their hierarchical structure. The solution to this MDP problem is the optimal scheduling policy that minimizes the average DFR.

- Using compact state space representation and function approximation, we simplified the MDP problem in order to solve it with RL. Our proposed RL scheduler is the solution given by an RL technique called SMART. Simulation results showed that the RL scheduler reduces the average DFR by 42%, 49%, and 53% when compared to EDD+SRPT [2], PAP [3], and F-SRPT [1] schedulers, respectively.

5.1 Future Work

Given that most of the current work in the literature related to scheduling in WPANs are either based on impulsed UWB or do not use application layer information, our work can pave the way for many future works. Due to deadlock in the standardization process of the physical layer for high-rate WPANs in IEEE 802.15 Task Group 3a (TG3a), the supporters of each scheme have developed their own standards for MAC. More specifically, the standard that is envisioned to be dominantly used for MAC in high-rate WPANs is ECMA-368 [58], which is partly similar to the IEEE 802.15.3 standard for MAC. Yet, because of some major differences, extending our work in the context of ECMA-368 MAC is a potential future work.

With the variety of applications that a high-rate WPAN should be able to support,

it will serve different classes of traffic with different QoS requirements. We focused on the video traffic class in this thesis. Therefore, integrating the FDA technique and/or the RL scheduler with a more versatile scheduler that can handle and recognize different traffic types can be a possible extension to our work.

Moreover, the issue of fairness among the scheduled video flows is also a critical point, which is not considered in our work. A possible future work is to account for fairness by formulating the problem as an MDP with constraints that enforce the fairness criteria. Using fuzzy logic for function approximation can also be of interest.

Bibliography

[1] R. Mangharam, M. Demirhan, R. Rajkumar, and D. Raychaudhuri, "Size matters: size-based scheduling for MPEG-4 over wireless channels," in *Proc. of SPIE/ACM Multimedia Computing and Networking (MMCN)*, Santa Clara, CA, Jan. 2004, pp. 110–122.

[2] A. Torok, L. Vajda, A. Vidacs, and R. Vida, "Techniques to improve scheduling performance in IEEE 802.15.3 based ad hoc networks," in *Proc. of IEEE Globecom*, St. Louis, MO, Nov. 2005.

[3] S. M. Kim and Y. J. Cho, "Scheduling scheme for providing QoS to real-time multimedia traffics in high-rate wireless PANs," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 4, pp. 1159–1168, Nov. 2005.

[4] "IEEE P802.15-03/268r3, Multi-Band OFDM Physical Layer Proposal for IEEE 802.15 Task Group 3a, Mar. 2004. Project: IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs)."

[5] "IEEE Std 802.15.3-2003: Wireless medium access control (MAC) and physical layer (PHY) specifications for high rate wireless personal area networks (WPANs)," Sept. 2003.

[6] S. Moradi and V. W. S. Wong, "Technique to improve MPEG-4 traffic schedulers in IEEE 802.15.3 WPANs," in *Proc. of IEEE International Conference on Communications (ICC)*, Glasgow, Scotland, June 2007.

[7] S. Moradi, V. W. S. Wong, and A. H. Mohsenian, "A novel scheduling algorithm for video traffic in high-rate WPANs," *submitted to IEEE Global Telecommunications Conference (GLOBECOM)*, Nov. 2007.

[8] X. Gu and L. Taylor, "Ultra-wideband and its capabilities," vol. 21, no. 3, pp. 56–66, July 2003.

[9] R. Cardinali, L. D. Nardis, M. G. Benedetto, and P. Lombardo, "UWB ranging accuracy in high- and low-data-rate applications," *IEEE Transactions on Microwave Theory and Techniques*, vol. 54, no. 4, pp. 1865–1875, June 2006.

[10] G. R. Aiello and G. D. Rogerson, "Ultra-wideband wireless systems," *IEEE Microwave Magazine*, vol. 4, no. 2, pp. 36–47, June 2003.

[11] X. Shen, W. Zhuang, H. Jiang, and J. Cai, "Medium access control in ultra-wideband wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 54, no. 5, pp. 1663–1677, Sept. 2005.

[12] A. Batra, J. Balakrishnan, G. R. Aiello, J. R. Foerster, and A. Dabak, "Design of a multiband OFDM system for realistic UWB channel environments," *IEEE Transactions on Microwave Theory and Techniques*, vol. 52, no. 9, pp. 2123–2138, Sept. 2004.

[13] J. H. Reed, *An introduction to ultra wideband communication systems.* Prentice Hall, Apr. 2005.

[14] "MPEG-4 part 2: Visual (IS 14496-2), doc. N2502," Oct. 1998.

[15] A. Ziviani, B. E. Wolfinger, J. F. Rezende, O. C. Duarte, and S. Fdida, "Joint adoption of QoS schemes for MPEG streams," *Multimedia Tools and Applications*, vol. 26, no. 1, pp. 59–80, May 2005.

[16] C. Y. Zou and Z. Haas, "Optimal resource allocation for UWB wireless ad hoc networks," in *Proc. of IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Berlin, Germany, Sept. 2005, pp. 452–456.

[17] J. Y. LeBoudec, R. Merz, B. Radunovic, and J. Widmer, "A MAC protocol for UWB very low power mobile ad hoc networks based on dynamic channel coding with interference mitigation," EPFL-DI-ICA, Tech. Rep. IC/2004/02, Jan. 2004.

[18] B. Radunovic and J. Y. L. Boudec, "Optimal power control, scheduling, and routing in UWB networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 7, pp. 1252–1270, Sept. 2004.

[19] Y. C. Chu and A. Ganz, "Joint scheduling and resource control for QoS support in UWB-based wireless networks," in *Proc. of IEEE Military Communications Conference (MILCOM)*, Monterey, CA, Nov. 2004, pp. 1100–1106.

[20] J. Shen, I. Nikolaidis, and J. Harms, "Dynamic 802.15.3 WPAN scheduling using maximal matching," in *Proc. of IEEE Global Telecommunications Conference (GLOBECOM)*, San Francisco, CA, Nov. 2006.

[21] F. Cuomo, C. Martello, A. Baiocchi, and F. Capriotti, "Radio resource sharing for ad hoc networking with UWB," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 9, pp. 1722–1732, Dec. 2002.

[22] K. H. Liu, L. Cai, and X. Shen, "Performance enhancement of medium access control for UWB WPAN," in *Proc. of IEEE Global Telecommunications Conference (GLOBECOM)*, San Francisco, CA, Nov. 2006.

[23] H. Jiang, K. H. Liu, and X. S. W. Zhuang, "Distributed medium access control in pulse-based time-hopping UWB wireless networks," in *Proc. of IEEE Global Telecommunications Conference (GLOBECOM)*, San Francisco, CA, Nov. 2006.

[24] L. Schrage, "Proof of the optimality of the shortest remaining processing time discipline," *Operations Research*, vol. 16, pp. 678–690, 1968.

[25] X. Liu, Q. H. Dai, and Q. F. Wu, "Scheduling algorithms analysis for MPEG-4 traffic in UWB," in *Proc. of IEEE Vehicular Technology Conference (VTC-Fall)*, Los Angeles, CA, Sept. 2004, pp. 5310–5314.

[26] S. S. Panwar, D. Towsley, and J. K. Wolf, "Optimal scheduling policies for a class of queues with customer deadlines to the beginning of service," *Journal of the ACM*, vol. 35, no. 4, pp. 832–844, 1988.

[27] E. Kwon, D. Hwang, and J. Lim, "An idle timeslot reuse scheme for IEEE 802.15.3 high-rate wireless personal area networks," in *Proc. of IEEE Vehicular Technology Conference (VTC-Fall)*, Dallas, TX, Sept. 2005, pp. 715–719.

[28] R. Zeng and G. S. Kuo, "A novel scheduling scheme and MAC enhancements for IEEE 802.15.3 high-rate WPAN," in *Proc. of IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, CA, Mar. 2005, pp. 2478–2483.

[29] S. H. Rhee, K. Chung, Y. Kim, W. Yoon, and K. S. Chang, "An application-aware MAC scheme for IEEE 802.15.3 high-rate WPAN," in *Proc. of IEEE Wireless Communications and Networking Conference (WCNC)*, Atlanta, GA, Mar. 2004, pp. 1018–1023.

[30] X. Chen, Y. Xiao, Y. Cai, J. Lu, and Z. Zhou, "An energy Diffserv and application-aware MAC layer scheduling for multiple VBR video streaming over high-rate WPANs," *Elsevier Computer Communications*, vol. 29, no. 17, pp. 3516–3526, Nov. 2006.

[31] Y. Xiao, X. Shen, and H. Jiang, "Optimal ACK mechanisms of the IEEE 802.15.3 MAC for ultra-wideband systems," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 4, pp. 836–842, Apr. 2006.

[32] G. Bianchi, "IEEE 802.11-saturation throughput analysis," *IEEE Communications Letters*, vol. 2, no. 12, pp. 318–320, Dec. 1998.

[33] X. Liu, Q. H. Dai, and Q. F. Wu, "Adaptive delayed acknowledgement algorithm for MPEG-4 traffic in UWB networks," *Tsinghua Science and Technology*, vol. 11, no. 3, pp. 278–286, June 2006.

[34] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming.* Wiley-Interscience, 1994.

[35] L. P. Kaelbling, M. L. Littman, and A. P. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[36] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* Cambridge, MA: MIT Press, 1998.

[37] A. Gosavi, "Reinforcement learning for long-run average cost," *European Journal of Operational Research*, vol. 155, pp. 654–674, 2004.

[38] T. K. Das, A. Gosavi, S. Mahadevan, and N. Marchalleck, "Solving semi-markov decision problems using average reward reinforcement learning," *Journal of Management Science*, vol. 45, no. 4, pp. 560–574, 1999.

[39] H. Robbins and S. Monro, "A stochastic approximation method," *Annals Mathematical Statistics*, vol. 22, pp. 400–407, 1951.

[40] A. Gosavi, N. Bandla, and T. K. Das, "A reinforcement learning approach to a single leg airline revenue management problem with multiple fare classes and overbooking," *IIE Transactions*, vol. 34, pp. 729–742, 2002.

[41] F. Yu, V. Wong, and V. Leung, "A new QoS provisioning method for adaptive multimedia in cellular wireless networks," in *Proc. of IEEE Infocom*, Hong Kong, China, Mar. 2004, pp. 2130–2141.

[42] A. Gosavi, "An algorithm for solving semi-markov decision problems using reinforcement learning: convergence analysis and numerical results," Ph.D. dissertation, University of South Florida, 1999.

[43] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing: Explorations in the microstructures of cognition.* Cambridge, MA: MIT Press, 1986, vol. 1.

[44] C. Y. Liao, F. Yu, V. Leung, and C. J. Chang, "Reinforcement-learning-based self-organization for cell configuration in multimedia mobile networks," *European Trans. Telecommunications*, vol. 16, pp. 385–397, 2005.

[45] J. S. Albus, *Brains, Behavior, and Robotics.* Peterborough, New Hampshire: BYTE Books, Subsidiary of McGraw Hill, 1981.

[46] R. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," *Advances in Neural Information Processing Systems*, vol. 8, pp. 1038–1044, 1996.

[47] A. W. Moore and C. G. Atkeson, "The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces," *Machine Learning*, vol. 21, no. 3, pp. 199–233, 1995.

[48] J. C. Santamaria, R. S. Sutton, and A. Ram, "Experiments with reinforcement learning in problems with continuous state and action spaces," *Adaptive Behavior*, vol. 6, no. 2, pp. 163–218, 1998.

[49] J. Gross, J. Klaue, H. Karl, and A. Wolisz, "Cross-layer optimization of OFDM transmission systems for MPEG-4 video streaming," *Computer Communications: Applications and Services in Wireless Networks*, vol. 27, no. 11, pp. 1044–1055, July 2004.

[50] F. Fitzek and M. Reisslein, "MPEG-4 and H.263 video traces for network performance evaluation," *IEEE Network*, vol. 15, pp. 40–54, 2001.

[51] J. M. Gimenez-Guzman, J. Martinez-Bauset, and V. Pla, "An afterstates reinforcement learning approach to optimize admission control in mobile cellular networks," *Lecture Notes in Computer Science: Wireless Systems and Network Architectures in Next Generation Internet*, vol. 3883, pp. 115–129, 2006.

[52] P. Kanerva, *Sparse distributed memory.* Cambridge, MA: MIT Press, 1988.

[53] B. Ratitch, S. Mahadevan, and D. Precup, "Sparse distributed memories in reinforcement learning: Case studies," in *Proc. of the Workshop on Learning and Planning in Markov Processes - Advances and Challenges*, San Jose, CA, July 2004, pp. 85–90.

[54] K. Kostiadis and H. Hu, "KaBaGe-RL: Kanerva-based generalisation and reinforcement learning for possession football," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Maui, HI, Nov. 2001, pp. 292–297.

[55] B. Ratitch and D. Precup, "Sparse distributed memories for on-line value-based reinforcement learning," in *Proc. of the European Conference on Machine Learning (ECML)*, Pisa, Italy, Sept. 2004, pp. 347–358.

[56] R. S. Sutton and S. D. Whitehead, "Online learning with random representations," in *Proc. of the International Conference on Machine Learning*, Amherst, MA, June 1993, pp. 314–321.

[57] C. Darken, J. Chang, and J. Moody, "Learning rate schedules for faster stochastic gradient search," in *Proc. of IEEE Workshop on Neural Networks for Signal Processing*, Copenhagen, Denmark, Sept. 1992.

[58] "ECMA-368 standard: High rate ultra wideband PHY and MAC standard," Dec. 2005.