An RL-based Scheduling Algorithm for Video Traffic in High-rate Wireless Personal Area Networks

Shahab Moradi, A. Hamed Mohsenian-Rad, and Vincent W.S. Wong
Department of Electrical and Computer Engineering
The University of British Columbia, Vancouver, Canada
e-mail: {shahab, hamed, vincentw}@ece.ubc.ca

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 dominant 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. In this paper, we first propose a Markov decision process (MDP) model for optimal scheduling for video flows in high-rate WPANs. Using this model, we also propose a scheduler that incorporates compact state space representation, function approximation, and reinforcement learning (RL). Simulation results show that our proposed RL scheduler achieves nearly optimal performance and performs better than F-SRPT, EDD+SRPT, and PAP scheduling algorithms in terms of a lower decoding failure rate.

Keywords: Wireless personal area networks, scheduling, QoS, ultra-wide band (UWB), Markov decision process (MDP), reinforcement learning.

This paper was presented in part in *IEEE Global Telecommunications Conference (Globecom)*, Washington, DC, December 2007.

I. Introduction

In recent 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. This makes UWB be 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 universal serial bus (USB).

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 [1], which is designed for WPANs, details of scheduling and QoS support are left to the developers. Consequently, in this paper, we aim to design an application-aware scheduling algorithm for MAC layer to provide the required QoS for video traffic.

Similar to other real-time traffic, video flow 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 [2].

Recently, various MAC scheduling algorithms have been proposed for high-rate WPANs (e.g., [2]–[10]). For impulse-based UWB, scheduling problems can be formulated as rate and power allocation problems. These problems can be modeled as a joint optimization problem, so as to minimize the total power consumption [7] or maximize the total system throughput [9]. The concept of exclusion region is also used for such schedulers [10]. On the other hand, with no assumption on the type of physical layer, Mangharam *et al.* proposed the fair shortest remaining

processing time (F-SRPT) scheduler [5]. SRPT schedules different jobs in the system in the order of their remaining processing time, from the shortest to the longest. F-SRPT is a variation of SRPT that maintains fairness among flows with different data rates. In [6], the earliest due date (EDD) method is used along with SRPT. In [11], Rhee *et al.* used the application layer information at MAC layer. Each device informs the piconet controller of the maximum size of its I, P, and B frames for video flows. The channel time is then allocated based on these values. Kim and Cho proposed a scheduling algorithm designed for MPEG-4 flows [2]. Each MPEG-4 frame type is scheduled with a pre-assigned priority (PAP) in I, P, and B order. The scheme proposed in [8] focuses on reducing the average waiting time by using an $M/M/c$ queuing model for channel time allocation at the piconet coordinator. It also proposed a command aggregation scheme in order to reduce MAC overhead when sending short command frames. Energy efficiency is considered by the scheduler proposed in [12]. 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. In [13], we proposed a frame-decodability aware (FDA) technique to improve the performance of scheduling of video flows.

The objective of our work is to design a systematic scheduling algorithm for video flows in high-rate WPANs. The contributions of in this paper are as follows:

- We provide a mathematical framework based on Markov decision process (MDP) to determine the optimal scheduler for video flows in high-rate WPANs. This framework takes into account the number and pattern of video flows, and their hierarchical structure.
- Using the MDP framework, along with compact state space representation, function approximation, and reinforcement learning (RL), we propose a practical scheduling algorithm which can provide significantly better QoS to video flows when compared to some other schedulers [14].
- Simulation results show that our proposed RL scheduler achieves nearly optimal perfor-

mance and performs 53%, 42%, and 49% better than F-SRPT, EDD+SRPT, and PAP schedulers, respectively [15].

The rest of this paper is organized as follows. In Section II, we provide an overview of the IEEE 802.15.3 standard for MAC, the hierarchical structure of video streams, related scheduling algorithms, the performance metrics for video flows, and reinforcement learning. In Section III, we describe the mathematical formulation for the optimal video scheduler. The implementation mechanisms of the RL algorithm are described in Section IV. Section V describes an extension of our proposed model. Results for performance evaluations and comparisons are presented in Section VI. Conclusions and future work are discussed in Section VII. A list of nomenclature is in Table I.

II. Background

A. IEEE 802.15.3 MAC Standard

The IEEE 802.15.3 is designed for WPANs and aims to provide low cost, low power consumption, and high data rate communications, within its area of operation called the piconet [1]. A piconet consists of a number of independent devices (DEVs) that communicate with each other under the control of the piconet coordinator (PNC). The PNC provides basic timing, performs scheduling, and manages the QoS requirements of the piconet.

Within a piconet, the timing is based on superframes, which consists of three parts (see Fig. 1). The first part, *beacon*, announces timing allocations, superframe duration, and other piconet synchronization parameters. The second part, the contention access period (CAP), is used to communicate commands and asynchronous data. The third part, channel time allocation period (CTAP), is composed of channel time allocations (CTAs) and management CTAs (MCTAs). A DEV can use its CTA, which is assigned to it by the PNC, for isochronous streams, asynchronous data transfer, or sending commands. Channel access during CTAP is TDMA and there is no

contention in this period. By sending beacon at the beginning of superframe, the PNC announces the start time and duration of each CTA as well as the DEVs that are allowed to use it.

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, and type and deadline of the frames in their queues, in order to allocate CTAs to different flows.

B. Video Streams

A typical video encoder generates a sequence of three types of compact frames: intra-coded (I), predictive (P), and bidirectional (B) frames [16]. Because of the different 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 [16]. This pattern is generally fixed for a given video sequence, and N is a multiple of M. Fig. 2 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* [17]. 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 lost. As an example, in Fig. 2, 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 II 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.

C. Scheduling Algorithms

In an 802.15.3 piconet, the CTA requirements of a flow can be regarded as a job. 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 [18]. 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 systems with heterogeneous flows that have different average data rates, SRPT favors the flows with smaller average data rate since they tend to have smaller frames. In [5], Mangharam *et al.* proposed a 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 [5] 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.

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 [19]. However, in 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. 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 may have equal deadlines, hence a tie-breaking method is needed.

In [6], Torok *et al.* used EDD along with SRPT as the tie-breaking method. They 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.

The pre-assigned priority (PAP) in [2] schedules video frames their importance (i.e., in I, P, and B order). EDD and SRPT are used to break ties among frames of the same type, when they cannot be served together in one superframe. The work in [2] used 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.

D. Performance Metrics: JFR and DFR

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. DFR is defined as the ratio of the total number of undecodable frames to the total number of frames [17]. DFR can be viewed as an objective measure of user-perceived degradation of quality, and is the main performance metric in this paper.

E. Reinforcement Learning (RL)

In many real-life sequential decision making problems, dynamic programming (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 [20]. In RL, an 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. In an RL model, the agent has an indication of the system state. The agent takes action, 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). The agent should choose actions that tend to increase some function of the reward. 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 [20] [21]. 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 [22].

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 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 [21].

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. 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.

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 [23]. As we will show in Section III-D, the average reward criterion is more suitable for our scheduling problem. We use an algorithm called SMART (Semi-Markov Average Reward Technique) [24] [25] [26] that is designed for average reward RL. The convergence analysis of this algorithm is given in [25] and it has been successfully applied to different problems including production inventory [24], airline seat allocation [26], and QoS provisioning in wireless cellular networks [23]. The difference between Semi-Markov decision process (SMDP) and Markov decision process (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 III we formulate the scheduling problem in the form of an MDP, we can use this algorithm with fixed state transition duration.

III. Optimal Scheduling Algorithm for Video Traffic

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 [27]. We then 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 [20] [21] to find the scheduling policy which minimizes average DFR.

Consider a WPAN and let F denote the total number of video flows. We denote $\mathcal{F} = \{1, 2, \ldots, F\}$ as the set of all flows. We assume that the deadline of all video frames is fixed[1], and is equal to the 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} video flow ($i \in \mathcal{F}$) 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. An MDP problem formulation includes the set of decision epochs, the set of system states, the set of actions, the reward function, and the state transition probabilities. We define these elements for our MDP in the following subsections.

A. 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

[1]Notice that a tight fixed deadline indicates the worst-case scenario in which the source codec has no buffering capability; thus, it can only handle one frame every γ (i.e., frame inter-arrival time) seconds.

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 the n^{th} superframe, is $(n-1)\eta$. Therefore, the set of decision epochs is $\{1, 2, \ldots, T\}$.

B. States

Let the function $\text{tx}_i(x)$ denote 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:

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

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 remaining until the arrival of the next frame of flow i. 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 of flow i. For the first frame of the GOP $g_i = 0$, and for the last one $g_i = N_i - 1$.
- $\delta \triangleq [\delta_1\ \delta_2\ \cdots\ \delta_F]$, and δ_i is a binary variable which denotes whether the current frame in the queue of flow i can be decoded successfully at the receiver. 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.

- $D^{max} \triangleq \lfloor \frac{\gamma}{\eta} \rfloor$, 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 [13], [14] to determine the value of δ based on $\mathbf{l}$, $\mathbf{d}$ and $\mathbf{g}$.

C. Actions

Let $\mathbf{A}$ be the set of all possible actions. We have,

$$\mathbf{A} = \left\{ \mathbf{a} \triangleq (a_1,\ a_2,\ \cdots\ a_F,\ p) \,|\, p \in \mathcal{F} \cap \{0\};\ \ a_i \in \{0,1\},\ \forall\ i \in \mathcal{F};\ a_p = 0 \text{ if } p \neq 0 \right\}, \quad (2)$$

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. To explain this, we notice that in general, we are *not* interested in partial transmissions as a half (incomplete) frame is not useful for a video decoder at the destination node. In fact, we will see that our introduced rewarding function in Section III-D only rewards *full* transmissions. Nevertheless, we still allow a single *long* (i.e., almost complete) partial transmission as it is more similar to a full transmission. The last conditional statement in (2) implies that the frame of flow p cannot be both fully and partially transmitted simultaneously.

At each decision epoch, 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:

$$\sum_{i=1}^{F} (a_i \cdot l_i) \leq \eta. \quad (3)$$

In addition, all the scheduled flows must be eligible. That is,

$$a_i \leq e_i, \qquad \forall\, i \in \mathcal{F} \tag{4}$$

$$d_p > 1 \text{ and } e_p = 1, \quad \text{if } p \neq 0, \tag{5}$$

where e_i denotes the *eligibility* of the flow i for being scheduled. Flow $i \in \mathcal{F}$ is eligible ($e_i = 1$) if it has a frame that is not expired ($d_i > 0$) and is decodable ($\delta_i = 1$); otherwise, it is ineligible for being scheduled ($e_i = 0$). Equation (4) implies that $a_i = 0$, if $e_i = 0$. In other words, the ineligible flows are never scheduled. Equation (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:

$$l_i > l^{partial}, \qquad \forall\, i \in \mathcal{F},\ e_i = 1,\ a_i = 0. \tag{6}$$

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 $\mathbf{s} \in \mathbf{S}$, denoted by $\mathbf{A_s}$, is the largest subset of $\boldsymbol{A}$, whose members satisfy all the constraints (3)–(6). These constraints guarantee that the scheduler accommodates as many eligible frames as possible. Note that $\mathbf{A_s}$ is stationary and only depends on the system states.

D. Reward Function and Gain

As mentioned in Section II-B, 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 II shows the number of dependencies between GOP frames of an (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:

$$r(\mathbf{s}(n), \mathbf{a}(n)) = \sum_{i=1}^{F} \Big[a_i(n) - c_i(\mathbf{s}(n))(1 - a_i(n)) \Big]. \tag{7}$$

In order to show the merits of the reward function in (7), we study the policy gain that it yields. The gain of policy π under the average reward criterion is the average accumulated reward. In our formulation, the policy gain is given by:

$$\begin{aligned} \rho^{\pi} &= \frac{1}{T} \sum_{n=1}^{T} r(\mathbf{s}(n), \mathbf{a}(n)) \\ &= \sum_{i=1}^{F} \left(\frac{\sum_{n=1}^{T} a_i(n) - \sum_{n=1}^{T} c_i(\mathbf{s}(n))(1 - a_i(n))}{T} \right). \end{aligned} \tag{8}$$

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 (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 (8) can be rewritten as:

$$\begin{aligned}
\rho^{\pi} &= \frac{\eta}{\gamma}\sum_{i=1}^{F}\Big(\frac{\text{total_scheduled} - \text{total_undecodable}}{\text{total_frames}}\Big) \\
&= \frac{\eta}{\gamma}\sum_{i=1}^{F}\Big(1 - \frac{2\times \text{total_undecodable}}{\text{total_frames}}\Big) \\
&= \frac{\eta}{\gamma}\sum_{i=1}^{F}(1 - 2\text{DFR}_i^{\pi}) \\
&= \frac{\eta F}{\gamma} - \frac{2\eta}{\gamma}\sum_{i=1}^{F}\text{DFR}_i^{\pi}, \qquad (9)
\end{aligned}$$

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 (9), we have

$$\overline{\text{DFR}}^{\pi} = \frac{1}{2} - \frac{\gamma}{2\eta F}\rho^{\pi}. \qquad (10)$$

Equation (10) shows that in our formulation, the average DFR is a linear function of gain. Therefore, we conclude that:

$$\arg\max_{\pi}\rho^{\pi} = \arg\min_{\pi}\overline{\text{DFR}}^{\pi}. \qquad (11)$$

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

E. 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 relate 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. Thus, 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.

Let $\text{prob}_i(l_i^{new}(n))$ denote the probability that the channel time request of flow i is $l_i^{new}(n)$.

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

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

This probability is simply related to the frame size distribution probability of flow i.

IV. Algorithm Implementation: RL Scheduler

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 dynamic programming 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 [24] to overcome the curse of modeling. Second, in the state space defined in equation (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, $|\mathbf{S}| = (8 \times 5 \times 12 \times 2)^F \approx 1000^F$. Furthermore, it is easy to verify that $|\mathbf{A}| = F \cdot 2^F$. The semi-Markov average reward techique (SMART) learns the Q-function $Q : \mathbf{S} \times \mathbf{A} \rightarrow \mathbb{R}$, which has impractical memory requirement and convergence rate given the size its domain. To tackle this issue, we solve the problem in two phases. In Section IV-A, we describe a compact representation of the state-action space, which is a mapping from $\mathbf{S} \times \mathbf{A}$ to a smaller set $\mathbf{\Omega}$. Then, in Section IV-B, a memory-based function approximator is used to map $\mathbf{\Omega}$ to the set of real numbers $\mathbb{R}$.

A. Features and State Space Representation

We use an aggregation scheme [21], 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 [21]. We refer to the system state *after* the effect of the taken action as *afterstate* [21]. 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 $\mathbf{S} \times \mathbf{A}$ to $\mathbf{S}$. Reference [28] successfully applies this idea to admission control problem in cellular networks.

The state space of our MDP has binary variables in δ, 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 $\mathbf{S}$, 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:

- $F_{y,\,non\_urgent} = \left|\left\{i \mid l_i > 0,\ y_i(g_i) = y,\ \delta_i = 1,\ d_i > 1\right\}\right|$ is the number of all the frames of type y in the system that have a deadline 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 $\mathbf{S} \times \mathbf{A}$ to $\mathbf{S}$. The second aggregation is feature extraction, which is a function from $\mathbf{S}$ to $\mathbf{\Omega}$. Using the procedures described in this subsection, we can define the function $f\colon \boldsymbol{S} \times \boldsymbol{A} \rightarrow \mathbf{\Omega}$, 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 $\mathbf{\Omega}$.

B. 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 [21]. 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 [21]. It was originally presented in the context of sparse distributed memory [29]. 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 [30] and [31].

In our scheduling problem, we use Kanerva coding to approximate the function $\hat{Q}\colon \mathbf{\Omega} \rightarrow \mathbb{R}$, which gives the value of a feature vector. The prototypes of Kanerva coding are feature vectors in $\mathbf{\Omega}$. The value of any feature vector in $\mathbf{\Omega}$ 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 $\mathbf{x}, \mathbf{z} \in \mathbf{\Omega}$ as the number of unequal features in the two vectors, i.e., $\text{dist}(\mathbf{x}, \mathbf{z}) \triangleq \sum_{j=1}^{|\mathbf{x}|} U_{\{x_j \neq z_j\}}$ [31]. Furthermore, we define the similarity of two feature vectors $\mathbf{x}$ and $\mathbf{z}$ as follows [32]:

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

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

$$\hat{Q}(\mathbf{x}) = \frac{\sum_{k \in H_{\mathbf{x}}} \lambda_k w_k}{\sum_{k \in H_{\mathbf{x}}} \lambda_k}. \tag{12}$$

Thus, the function $\hat{Q}(\cdot)$ is a linear combination of the value of prototypes. Using the standard gradient descent algorithm [21] for linear approximation, the update equation becomes [32]:

$$w_i \leftarrow w_i + \mu \frac{\lambda_i}{\sum_{k \in H_{\mathbf{x}}} \lambda_k} \left\{ \left[r(\mathbf{s}, \mathbf{a}) - \eta\rho + \max_{\mathbf{a}'} \hat{Q}(f(\mathbf{s}', \mathbf{a}')) \right] - \hat{Q}(f(\mathbf{s}, \mathbf{a})) \right\}, \; \forall \; i \in H_{\mathbf{x}}. \tag{13}$$

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 [32]. 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:

$$\lambda(\mathbf{h}_i, \mathbf{h}_j) \leq \begin{cases} 1 - \frac{1}{K-1}, & K \geq 3 \\ 0.5, & K = 2 \end{cases} \tag{14}$$

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 (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 (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 (12).

For discussion on how random selection of prototypes can give good approximation, see [33]. Successful application of this algorithm to RL tasks can be found in [30].

C. RL Scheduler

Since we use average reward optimality criterion, SMART [24] 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 Fig. 3. 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 [34]. 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 in Fig. 3, the RL scheduler executes the chosen action and calculates the immediate reward (see equation (7)) and the next system state $\mathbf{s}'$. In Step 4, the scheduler updates the set of prototypes according to Rule 1 and Rule 2 of Section IV-B. The weights of the prototypes are updated in Step 5. This update is the same as equation (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. Fig. 4 illustrates the structure of our proposed RL scheduler.

V. Model Extension

In the previous subsections, we assume that the deadline of all video frames is fixed, and is equal to the inter-arrival time of a frame, i.e., γ. In this section, we describe how to extend our

proposed model and relax the fixed deadline assumption. We consider the general case where the deadline for each flow i is equal to $\Gamma_i \geq \gamma$. In particular, we assume that:

$$\Gamma_i = J_i\gamma, \qquad \forall\, J_i \in \{1, 2, \ldots\},\ i = 1, \ldots, F. \tag{15}$$

Clearly, this require the queue of flow i needs to hold up to J_i frames. To incorporate the extended deadline model in (15) into our problem formulation, we need to modify the state, action, and reward models in our RL scheduler design. First, consider the set of system states in (1). If (15) holds, then set $\mathbf{S}$ becomes:

$$\begin{aligned}\mathbf{S} \ \triangleq\ \Big\{\mathbf{s} = (\mathbf{l}, \mathbf{d}, \mathbf{g}, \delta)\,|\,\forall\, i \in \mathcal{F}, 1 \leq j \leq J_i,\ \ 0 \leq l_{i,j} \leq \mathrm{tx}_i(L_i^{max}),\\ d_{i,j} \in \{0, \ldots, D_i^{max}\},\ g_{i,j} \in \{0, \ldots, N_i - 1\},\ \delta_{i,j} \in \{0, 1\}\Big\}.\end{aligned} \tag{16}$$

Here $\mathbf{l} \triangleq [l_{1,1} \cdots l_{1,J_1}\ l_{F,1} \cdots l_{F,J_F}]$, where $l_{i,j}$ denotes the channel time required by the j^{th} frame in the queue of flow i, in milliseconds. the vectors $\mathbf{d}$, $\mathbf{g}$, and δ can be obtained similarly. On the other hand, the value $D_i^{max} \triangleq \lfloor \frac{\Gamma_i}{\eta} \rfloor = \lfloor \frac{\gamma J_i}{\eta} \rfloor$, is the maximum frame deadline in units of superframe size η.

Similarly, we can extend the set of all possible actions to be as follows:

$$\begin{aligned}\mathbf{A} \ =\ \Big\{\mathbf{a} \triangleq (a_{1,1},\ \cdots\ a_{F,J_F},\ p, q)\,|\ \ p \in \mathcal{F} \cap \{0\};\ 1 \leq q \leq J_i\ a_{i,j} \in \{0, 1\},\ \forall\, i \in \mathcal{F},\\ j \in \{1, \ldots, J_i\};\ a_{pq} = 0 \text{ if } p \neq 0\Big\},\end{aligned} \tag{17}$$

where a_{ij} is equal to 1 if the scheduler allocates enough channel time to flow i so that it can *fully* transmit its j^{th} frame. Given the action set $\mathbf{A}$, at each epoch, the extended scheduler should choose an action depending on the current state. Clearly, the channel time required to transmit all scheduled frames for all flows should not exceed the length of the superframe. This requires replacing (3) with the following extended constraint:

$$\sum_{i=1}^{F}\sum_{j=1}^{J_i}(a_{i,j} \cdot l_{i,j}) \leq \eta. \tag{18}$$

We notice that if $\Gamma_i = \gamma$ for all flows $i = 1, \ldots, F$, then the constraint in (18) simply reduces to (3). After transmitting all the scheduled frames, the remaining time can be calculated as:

$$l^{partial} = \eta - \sum_{i=1}^{F} \sum_{j=1}^{J_i} (a_{i,j} \cdot l_{i,j}) \leq \eta. \tag{19}$$

Next, consider the reward model. We can write the state-dependent cost function as follows:

$$c_{i,j}(\mathbf{s}) = \begin{cases} \big(N_i + (M_i - 1)\big) \cdot e_{i,j} \cdot U_{\{d_{i,j}=1\}}, & \text{if } y_i(g_{i,j}) = I, \\ e_{i,j} \cdot U_{\{d_{i,j}=1\}}, & \text{if } y_i(g_{i,j}) = B, \\ \big(N_i - 1 - (\frac{g_{i,j}}{M_i} - 1)M_i\big) \cdot e_{i,j} \cdot U_{\{d_{i,j}=1\}}, & \text{if } y_i(g_{i,j}) = P. \end{cases} \tag{20}$$

Note that $y_i(\ )$, M_i and N_i do *not* have j index because they are indeed defined in per flow basis. We can also extend the reward function to be as follows:

$$r(\mathbf{s}(n), \mathbf{a}(n)) = \sum_{i=1}^{F} \sum_{j=1}^{J_i} \Big[a_{i,j}(n) - c_{i,j}(\mathbf{s}(n))(1 - a_{i,j}(n)) \Big]. \tag{21}$$

Given the reward function above, we can rewrite the policy gain to be:

$$\begin{aligned} \rho^{\pi} &= \frac{1}{T} \sum_{n=1}^{T} r(\mathbf{s}(n), \mathbf{a}(n)) \\ &= \sum_{i=1}^{F} \left(\frac{\sum_{n=1}^{T} \sum_{j=1}^{J_i} a_{i,j}(n) - \sum_{n=1}^{T} \sum_{j=1}^{J_i} c_{i,j}(\mathbf{s}(n))(1 - a_{i,j}(n))}{T} \right). \end{aligned} \tag{22}$$

In this case, 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 $\sum_{n=1}^{T} \sum_{j=1}^{J_i} a_{i,j}(n)$ and $\sum_{n=1}^{T} \sum_{j=1}^{J_i} c_{i,j}(\mathbf{s}(n))(1 - a_i(n))$ in (22) 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, after reordering the terms, the policy gain in (22) becomes exactly the same as that in (9). Therefore, (10) also still holds and an optimal (maximum) gain policy yields the optimal (minimum) average DFR, which is what we aim to find in presence of the extended model.

Next, we extend the state transitions model to the general case when the deadline Γ_i for flow i is as in (15). At superframe n, the system is in state $\mathbf{s}(n)$ and chooses the action $\mathbf{a}(n)$. The next

state $\mathbf{s}(n+1)$ 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 relate to each flow change. The whole system state is updated by performing the same procedure for all the flows.

When $d_{i,1}(n) \geq 1$, no new video frame will arrive for flow i. Thus, the remaining time until the next arrival is reduced by one superframe. If any frame of a 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 $\delta_{i,j}$ 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 $d_{i,1}(n) \geq 1$. Hence, if $d_{i,1}(n) \geq 1$, then for each $j \in \{1 \dots J_i\}$, the state *deterministically* changes as follows:

$$d_{i,j}(n+1) = d_{i,j}(n) - 1, \tag{23}$$

$$l_{i,j}(n+1) = l_{i,j}(n)(1 - a_{i,j}(n)) - l^{partial} U_{\{i=p \text{ and } j=q\}}, \tag{24}$$

$$g_{i,j}(n+1) = g_{i,j}(n), \tag{25}$$

$$\delta_{i,j}(n+1) = \delta_{i,j}(n). \tag{26}$$

When $d_{i,1}(n) = 0$, it means that the first frame in the queue of flow i has expired, if it has not already been sent, i.e., if $l_{i,1}(n) \neq 0$. It also means that a new frame will arrive for flow i, within superframe n. We shift the frames in the queue (getting rid of the head frame) and add the new frame to the tail of the queue. 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.

Let $\text{prob}_i(l_i^{new}(n))$ denote the probability that the channel time request of flow i is $l_i^{new}(n)$. Thus, when $d_{i,1}(n) = 0$,the state changes with probability $\text{prob}_i(l_i^{new}(n))$ as follows:

$$d_{i,J_i}(n+1) = d_i^{new}(n), \tag{27}$$

$$l_{i,J_i}(n+1) = l_i^{new}(n), \tag{28}$$

$$g_{i,J_i}(n+1) = (g_{i,J_i}(n) + 1) \bmod N_i. \tag{29}$$

We notice that the exact model for $\delta_{i,J}(n+1)$ depends on the considered scenario. It is very difficult (if possible) to provide a general closed-form model for $\delta_{i,J}(n+1)$ for extended deadline model. On the other hand, for each $j \in \{2, \cdots, J\}$ we have:

$$d_{i,j-1}(n+1) = d_{i,j}(n), \tag{30}$$

$$l_{i,j-1}(n+1) = l_{i,j}(n), \tag{31}$$

$$g_{i,j-1}(n+1) = g_{i,j}(n), \tag{32}$$

$$\delta_{i,j-1}(n+1) = \delta_{i,j}(n). \tag{33}$$

Given the extended models in (15)-(33), our proposed RL algorithm can be used to obtain the optimal solutions for the general case when the deadlines are indeed arbitrarily for each flow.

VI. Performance Evaluation and Comparisons

In this section, we evaluate the performance of our proposed scheduling algorithm. We use the real MPEG-4 trace of Jurassic Park movie with GOP pattern of ($N = 12$, $M = 3$) [35]. The frame rate is 30 frames per second. The average rate of each flow is 8 Mbps. In the simulation model, each iteration (i.e., simulation run) lasts 500 s. The superframe size is $\eta = 8$ ms; thus, each iteration consists of $500/0.008 = 62,500$ superframes (i.e., decision epochs). The parameters p_0 and μ_0 of the DCM scheme algorithm depend on how fast the scheduler can learn the optimal policy. We set $p_0 = \mu_0 = 0.1$ and $p_r = \mu_r = 10^{10}$, as it gives the RL scheduler enough time to explore and find the optimal policy without too many iterations. We end the simulation when both the learning rate and exploration probability fall below 0.005.

The rest of the simulation parameters are as follows. The channel data rate is 100 Mbps, the number of MPEG-4 flows varies from 2 to 10. The maximum tolerable delay for MPEG-4 frames is 1/30 s and maximum MAC fragment size is 2048 bytes. We use the average DFR as the performance metric. For F-SRPT [5], EDD+SRPT [6], and PAP [2] scheduling algorithms we use the performance that is already improved by the FDA technique described in [13].

Fig. 5 compares the average DFR achieved by RL, EDD+SRPT, PAP, and F-SRPT algorithms when the number of MPEG-4 flows varies from 2 to 10. We see that RL scheduler performs better than the others regardless of the number of MPEG-4 flows. The relative reduction of average DFR is up to 42%, 49%, and 53% for EDD+SRPT, PAP, and F-SRPT scheduler, respectively.

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 the start times of flows are separated by ϕ seconds. In other words, flow i starts at time $i\phi$ [5]. Fig. 6 compares the average DFR achieved by RL, F-SRPT, EDD+SRPT, and PAP scheduling algorithms when ϕ varies from 1 to 30 ms. The number of MPEG-4 flows is equal to 9. We can see that our proposed RL scheduler performs better than the other three for all values of ϕ. Furthermore, the performance of RL scheduler is less sensitive to ϕ.

To evaluate the optimality of the RL algorithm, we consider the special case of interest which is when $\phi = 0$, i.e., when all the flows in the system start at the same time. In [14], we show that for this case, the SRPT scheduler is the optimal scheduler. Fig. 7 compares the average DFR achieved by RL algorithm with the optimal case when the number of MPEG-4 flows varies from 2 to 10. We can see that in all cases, RL scheduler provides nearly optimal performance.

Table III 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 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 III, the policy gain and DFR have a linear relationship. We can verify the validity of equation (10) as follows. First, the estimated average DFR is calculated by substituting the policy gain ρ in equation (10). Second, the exact average DFR is measured by

counting the number of scheduled frames. Fig. 8 compares these two values. As one can see, $\overline{\text{DFR}}$ in equation (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 that the optimal gain policy yields the minimum average DFR.

VII. Conclusions and Future Work

In this paper, 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 when FDA technique is applied to the F-SRPT and EDD+SRPT schedulers, there are up to 61% and 60% reduction in DFR, respectively. Results also showed that the RL scheduler reduces the average DFR by 42%, 49%, and 53% when compared to EDD+SRPT, PAP, and F-SRPT schedulers, respectively.

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 paper. Therefore, extending the RL scheduler which can handle and recognize different traffic types is part of our work. This is possible by defining proper reward function and modifying the state space model. In the new setting, the scheduler can be designed to achieve the best *overall* performance based on a combination of different rewards and costs. Moreover, the issue of fairness among the scheduled video flows is also a critical point, which is not considered in this paper. Another direction for future work is to account for fairness by

formulating the problem as an MDP with constraints that enforce the fairness criterion. Using fuzzy logic for function approximation can also be of interest. Furthermore, it is possible to accommodate more flows into the system if we can take concurrent transmissions into account using *time domain spreading* (TDS) as well as *frequency domain spreading* (FDS) techniques. Finally, it is possible to extend our model to incorporate the impact of jitter by framing the arrival time probabilistically, rather than deterministically.

ACKNOWLEDGMENT

This work is supported by Bell Canada and the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

[1] "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.

[2] 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.

[3] 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.

[4] C. Hu, H. Kim, J. Hou, D. Chi, and S. Shankar, "Provisioning quality controlled medium access in UWB-operated WPANs," in *Proc. of IEEE Infocom*, Barcelona, Spain, Apr. 2006.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] S. Moradi, "A novel scheduling algorithm for video flows in high-rate WPANs," Master's thesis, University of British Columbia, Vancouver, Canada, Apr. 2007. [Online]. Available: http://www.ece.ubc.ca/∼vincentw/T/moradi.pdf

[15] S. Moradi, A. H. Mohsenian-Rad, and V. W. S. Wong, "A novel scheduling algorithm for video traffic in high-rate WPANs,," in *Proc. of IEEE Global Telecommunications Conference (Globecom)*, Washington, DC, Dec. 2007.

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

[17] 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.

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

[19] 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.

[20] 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.

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

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

[28] 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.

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

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

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

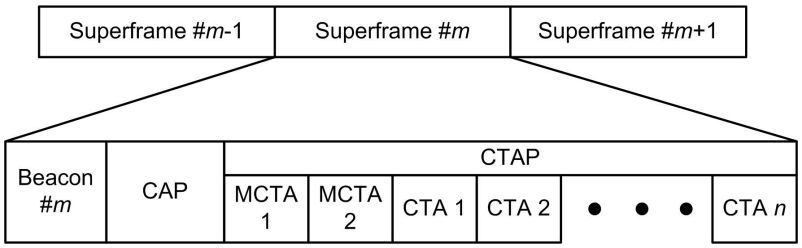

Fig. 1. Superframe structure. The presence of CAP is optional. The number, duration and order of the CTAs and MCTAs can vary from one superframe to another.

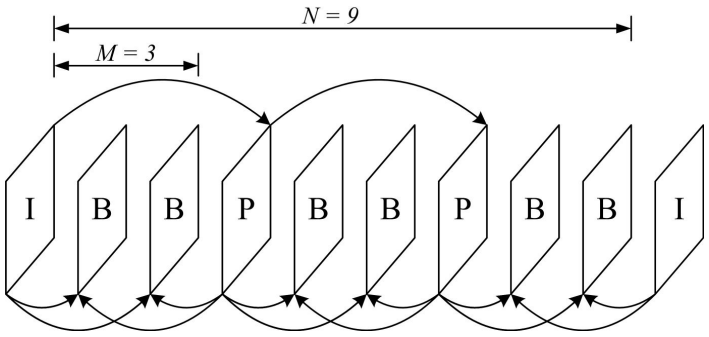

Fig. 2. GOP structure (N = 9, M = 3). The arrows indicate the direction of decoding dependencies.

TABLE I

LIST OF NOMENCLATURE.

Notation	Parameter definition
$\mathbf{A}$	Action set in MDP model.
$\mathbf{a}$	Action vector in MDP model.
a_i	Binary action for flow i.
B	Bidirectional frame.
D^{max}	maximum frame deadline in units of superframe size η.
d_i	Number of full superframes left until the arrival of the next frame.
e_i	A binary variable to denote the eligibility of flow i being scheduled.
F	Number of video flows.
$\mathcal{F}$	Set of video flows.
g_i	Offset of the frame in the queue of flow i, with respect to the beginning of the frame's GOP.
$\mathbf{h}_k$	The k^{th} prototype in Kanerva coding.
I	Intra-coded frame.
L_i^{max}	Maximum frame size of flow i.
l_i	Channel time required by the frame in the queue of flow i, in milliseconds.
l^{partial}	Remaining channel time.
M_i	I-to-P frame distance of the i^{th} video flow.
N_i	I-to-I frame distance of the i^{th} video flow.
P	Predictive frame.
p	The flow which can only transmit parts of its frame during a scheduling period.
$\mathbf{S}$	State space in MDP model.
$\mathbf{s}$	State vector in MDP model.
$\mathbf{\Omega}$	Set of feature vectors.
β	Activation radius in Kanerva coding.
δ_i	Binary variable for flow i.
η	Size of a superframe.
γ	Inter-arrival time.
$\lambda(\ )$	Similarity function.
ρ^{π}	Policy gain given the use of policy π.
$\text{tx}_i(x)$	Channel time that flow i requires to send x bits of data.

TABLE II

NUMBER OF FRAME DEPENDENCIES FOR EACH VIDEO FRAME

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

TABLE III

QUANTITATIVE COMPARISON AMONG RL SCHEDULER AND CONVENTIONAL SCHEDULERS. THE MINIMUM AVERAGE DFR GIVEN BY F-SRPT, EDD+SRPT, AND PAP IS USED FOR COMPARISON.

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

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 $\{\mathbf{A}_{\mathbf{s}} \setminus \mathbf{a}\}$.
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 IV-B 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 $\mathbf{s} \leftarrow \mathbf{s}'$.

Fig. 3. Pseudo-code of the RL scheduler.

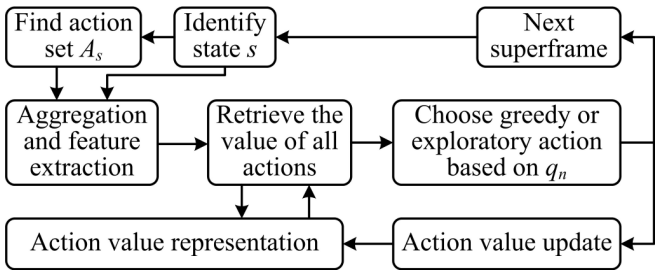

Fig. 4. Structure of the RL scheduler.

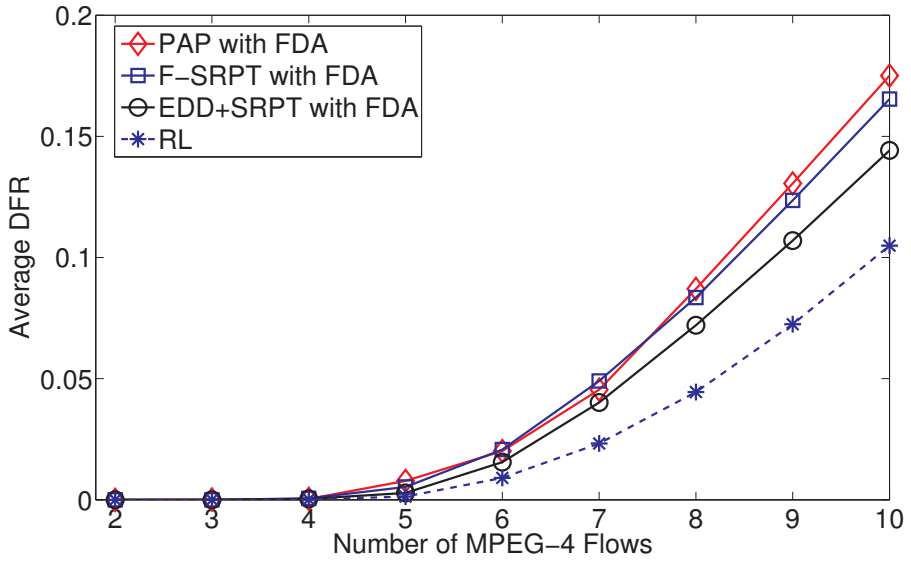

Fig. 5. Comparison of RL scheduler and other schedulers when the number of video flows varies from 2 to 10.

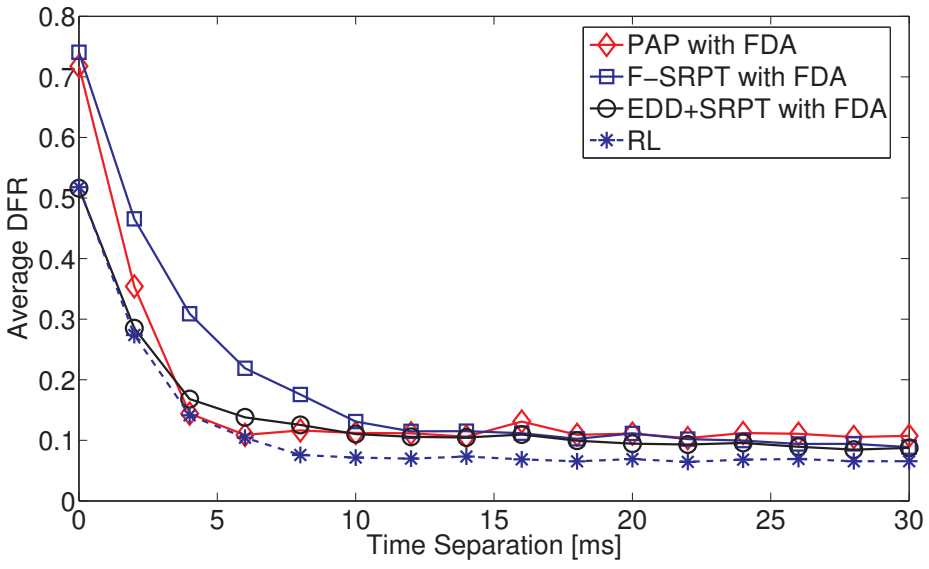

Fig. 6. Effect of time separation on average DFR. The RL scheduler has the smallest average DFR for all ϕ.

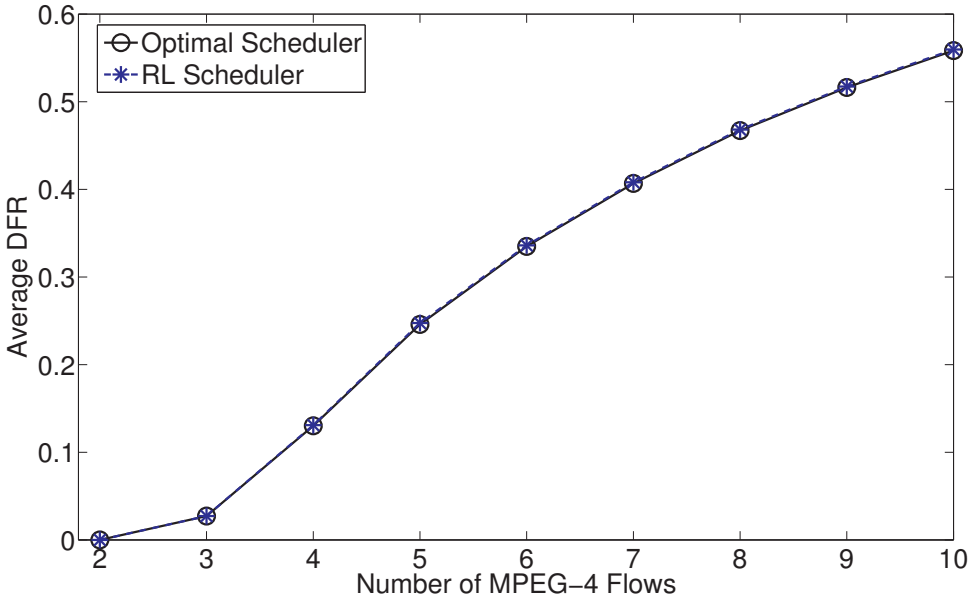

Fig. 7. Comparison of RL scheduler and optimal scheduler for $\phi = 0$. RL scheduler is nearly optimal in this case.

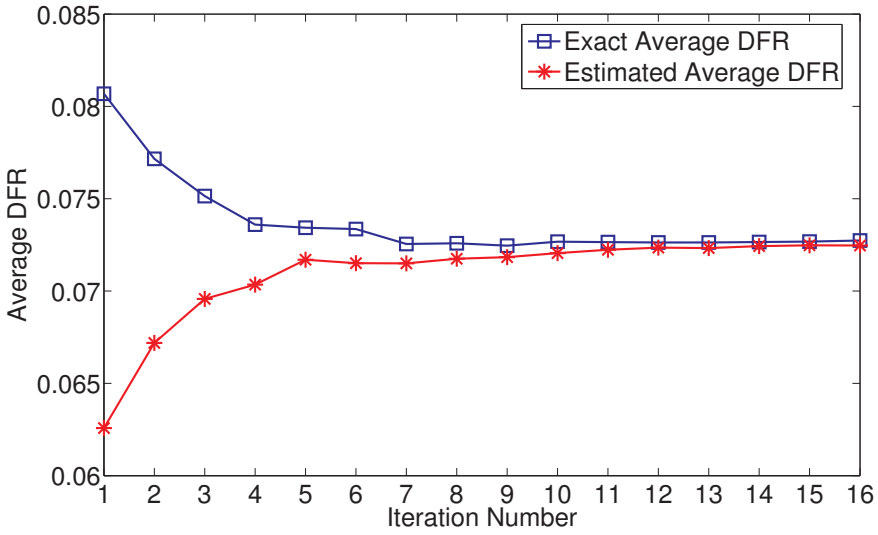

Fig. 8. Comparison of the exact and estimated DFR ($F = 9$).