Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2009, Article ID 482764, 12 pages
doi:10.1155/2009/482764

Research Article

QoS Differentiated and Fair Packet Scheduling in Broadband Wireless Access Networks

Rong Yu,[1] Yan Zhang,[2] and Shengli Xie[1]

[1] *School of Electronic and Information Engineering, South China University of Technology, Guangzhou 510641, China*
[2] *Simula Research Laboratory, 1325 Lysaker, Norway*

Correspondence should be addressed to Rong Yu, yurong@scut.edu.cn

Received 12 February 2009; Accepted 7 September 2009

Recommended by Thomas Michael Bohnert

This paper studies the packet scheduling problem in Broadband Wireless Access (BWA) networks. The key difficulties of the BWA scheduling problem lie in the high variability of wireless channel capacity and the unknown model of packet arrival process. It is difficult for traditional heuristic scheduling algorithms to handle the situation and guarantee satisfying performance in BWA networks. In this paper, we introduce learning-based approach for a better solution. Specifically, we formulate the packet scheduling problem as an average cost Semi-Markov Decision Process (SMDP). Then, we solve the SMDP by using reinforcement learning. A feature-based linear approximation and the Temporal-Difference learning technique are employed to produce a near optimal solution of the corresponding SMDP problem. The proposed algorithm, called Reinforcement Learning Scheduling (RLS), has in-built capability of self-training. It is able to adaptively and timely regulate its scheduling policy according to the instantaneous network conditions. Simulation results indicate that RLS outperforms two classical scheduling algorithms and simultaneously considers: (i) effective QoS differentiation, (ii) high bandwidth utilization, and (iii) both short-term and long-term fairness.

1. Introduction

With the rapid growth of demands on wireless multimedia applications and wireless data services, it is expected that broadband wireless access (BWA) networks should be able to support high-speed, high-quality, realtime, and nonrealtime applications, such as video, image, voice, text, and data. Quality of Service (QoS) provisioning for heterogeneous traffic with diverse properties is a necessity in emerging BWA networks. Packet scheduling algorithms, which are in charge of the bandwidth allocation and multiplexing at the packet level, play key roles for resource sharing and QoS provisioning in BWA networks.

Different from traditional wired scheduling problems, packet scheduling in BWA networks will encounter two major difficulties: the high variability of wireless channel conditions and the unknown model of packet arrival process. The capacity of wireless channel usually suffers either moderate or drastic fluctuations due to the signal attenuation, inter-user interference, fading and user mobility, and so forth. As a result, the channel conditions of users may vary with time and location asynchronously at random, leading to the so-called *time-dependent* error and *location-dependent* error [1]. In BWA networks, the process of packet arrivals is generally complicated and hard to accurately model in advance, since the packet arrivals are induced by the aggregation of multiple classes of traffic with different properties, such as constant and variable rates, and continuous and burst modes.

Besides the difficulties, packet scheduling algorithms for wireless multimedia networks are required to concurrently achieve three performance objectives: QoS differentiation and guarantee, wireless bandwidth utilization maximization, and fairness provisioning. Firstly, the transmission of heterogeneous classes of traffic with diverse QoS requirements naturally demands the scheduler to provide differential services. Secondly, the scheduler should be able to arrange the packet transmissions reasonably so that the precious wireless bandwidth can be utilized in an efficient way. Thirdly, the scheduler should also make sure that every traffic flow gets its fair share of the system resources. These three objectives, in most practical situations, are tightly coupled and complementary with each other. A good scheduling algorithm should provide satisfying performance tradeoff among them.

These special difficulties and performance objectives make the scheduling problem in BWA networks considerably challenging. The packet scheduler of a BWA network should operate across different sessions (i.e., connections or traffic flows) to guarantee that all sessions receive services with qualities according to the contract established when sessions are setup. At the beginning of each round of scheduling, the scheduler should decide one of the sessions for transmission. In order to make the right decision at each round of scheduling and hence accumulatively achieve the optimal long-term performance, the scheduler should be capable to predict the possible rewards (or costs) induced by different scheduling actions. However, the high variability of wireless channel conditions and the unknown characteristics of traffic flows make this long-term prediction difficult. Meanwhile, the multiple performance objectives introduce a set of interactional factors that should be considered together for prediction.

In this paper, we observe that the problem of packet scheduling in BWA networks could be described as a semi-Markov Decision Process (SMDP). There are two major advantages to solve the scheduling problem by the methodology of SMDP. Firstly, by modelling the scheduling problem as an SMDP, the entire scheduling process is divided into a series of stages. Decisions are made stage by stage. In other words, the original complicated problem is divided into multiple relatively simple subproblems, which are much more tractable than the original one. Secondly, there exist various of advanced mathematical techniques to solve the formulated SMDP problems, especially the learning-based approach. With the aid of learning-based approach, the scheduler has underlying ability to capture the main characteristics of the system and accurately predict the long-term rewards (or costs), even though there is no full knowledge of the system model.

The remainder of this paper is organized as follows. A brief introduction of existing packet scheduling algorithms is given in Section 2. The SMDP formulation is presented in Section 3. After introducing the system model, we cast the packet scheduling problem into the framework of SMDP. The construction of cost function is explained in details. Section 4 discusses the solution of the formulated SMDP problem. The methodology of reinforcement learning [2] is employed. A feature-based linear approximating architecture is adopted to approach the optimal average cost. The Temporal Difference (TD) technique [2] is used to continuously regulate the tunable parameter variables. Section 5 reports the results of simulation experiments, where the proposed algorithm is compared with two existing algorithms. Conclusion is presented in Section 6.

2. Related Work

In literature, the problem of packet scheduling is extensively studied in both wired and wireless environments. We broadly categorize the algorithms into three classes and present a brief introduction of them. For comprehensive overview of existing scheduling algorithms, readers are referred to [3].

2.1. Algorithms in Wired Environment. Algorithms of First-Come-First-Served (FCFS) and Round Robin (RR) are first developed for wired networks. Their original versions and improved versions (e.g., Weighted Round Robin and Deficit Round Robin [4]) are underlying schemes for wireless networks. Another well-known algorithm is Generalized Processor Sharing (GPS) [5], which is ideal and unimplementable in packet-based networks. Many other scheduling algorithms try to approximate GPS as far as possible. The merit of these wired scheduling algorithms is their simplicity and ease of implementation. But if directly used in BWA networks, these algorithms cannot achieve high bandwidth utilization and promise fairness guarantee.

2.2. Algorithms in Wireless Environment with GE-Model. The classical two-state Gilbert-Eilliot (GE) model [6, 7] is firstly used to model the wireless link variation. The channel is simply described to be either in “good” or in “bad” state in this model. A survey of this class of scheduling algorithms can be found in [1]. Work in [8] devised Idealized Weight Fair Queuing (IWFQ) algorithm. In [9], Channel-condition Independent Fair Queueing (CIF-Q) algorithm was put forward. Both algorithms of IWFQ and CIF-Q need to simulate a virtual error-free fair queueing system and try to schedule packets in the same order as the ideal reference system does. IWFQ and CIF-Q differ in the way they compensate the *lagging* flows. Wong et al. [10] presented Token Bank Fair Queueing (TBFQ) algorithm, which uses *token pools* and *token bank* to keep track of the service status of each flow and dynamically regulate the priorities of the flows to occupy the channel resource. All these three algorithms achieve good performance tradeoff among the three performance issues aforementioned. But they only work under the GE channel model, which is too coarse to characterize the practical variability of the wireless channel conditions.

2.3. Algorithms in Wireless Environment with Multistate Model. Some researchers considered variable-rate transmission in scheduling algorithms [11–15]. The Modified Largest Weighted Delay First (M-LWDF) algorithm [11] schedules packets according to a simple but effective constructed metric. The scheme was proved analytically to be *throughput optimal*. But it does not explicitly guarantee any fairness. The algorithm of Jalali et al. [12] was proposed to satisfy the so-called *proportional fairness*; the algorithm of Liu et al. [13] tried to maximize the long-term average total throughput for a given time fraction; the algorithm of Borst and Whiting [14] was made to be *Pareto-optimal*. The algorithm of Park et al. [15] selects user for transmission based on the cumulative distribution function (CDF) of the user rates.

Besides the above three classes of scheduling algorithms, researchers also consider standard-compatible schemes recently. In [16], a scheduling algorithm which is practical and compatible to the IEEE 802.16 standard is proposed. It provides QoS support in terms of bandwidth and delay bounds for all types of traffic classes as defined by the standard.

By summarizing the above-mentioned scheduling algorithms, we notice that most of the algorithms are designed based on heuristic methods. Either introducing the virtual/ideal service models as reference systems [8, 9] or defining some key metrics as criteria [11–14], many existing algorithms are built on the observation, judgement, intuition, and experience of the designers. Although heuristic algorithms have the merit of simplicity, their underlying drawbacks are twofold. Firstly, heuristic algorithms are generally proposed with respect to some given special outside conditions, which means that they are less of adaptivity. When the practical conditions do not match the provided ones, it is hard for heuristic algorithms to maintain their scheduling performance. Secondly, designing a heuristic scheduling algorithm is easy when the problem is relatively simple but could be rather tough when the scheduling environments and objectives become complicated. For packet scheduling problem in BWA networks, the scheduler is required to consider three main objectives simultaneously and, meanwhile, combat highly variable channels and unknown traffic models. Such a complicated scheduling problem is far out of the ability of a heuristic algorithm to handle.

To overcome the disadvantages of heuristic algorithms, we propose in this paper a learning-based algorithm for the packet scheduling problem in BWA networks. The proposed algorithm has in-built capacity of studying from outside environment and adapting system parameters according to the performance requirements. This attractive feature makes the proposed algorithm appropriate for the scheduling problem in BWA networks. More specifically, when this learning-based algorithm is applied in BWA networks, we only need to translate the scheduling objectives to the system reward (or cost) and then connect the scheduling problem with the ingredients of the solution framework. Little attention should be paid on the details of the entire scheduling process such as how to dynamically adapt the scheduler for the optimal performance. Because the scheduler has potential ability of self-training, it could automatically tuning itself towards an optimal system reward (or cost) according to the actual environment. Actually, the works in [13, 14] have realized that fixed and unadaptable scheduling schemes are incapable in complicated scheduling problem. Undetermined constants are integrated into the key metrics at the system initialization phase so that the scheduler could adjust its strategy at some extent. Nonetheless, the learning abilities of these algorithms are very limited, and they should still be looked on as heuristic algorithms.

3. Semi-Markov Decision Process Formulation

In this section, we cast the packet scheduling problem into the framework of Semi-Markov Decision Process (SMDP). As the base of our discussion, the wireless network model and channel model are firstly introduced. After that, we connect the packet scheduling problem with the ingredients of SMDP. The construction of cost function is comprehensively discussed.

3.1. System Model

3.1.1. Wireless Network Model. We consider a general shared-channel infrastructure-based BWA network, in which mobile hosts are served by a base station. The communication between a mobile host and the base station may contain more than one session (or traffic flow). Centralized scheduling of packet transmission is performed at the base station. A general scheduler model is described in Figure 1, where the usage of channel condition monitoring mechanism like LSM [17] is assumed. The method to implement LSM in actual systems can be found in [14]. In presence of LSM, the base station is supposed to have full knowledge of the channel conditions of all sessions.

3.1.2. Wireless Channel Model. The conditions of wireless links between the base station and each of the mobile hosts are independent of each other. We use the multistate Markov channel model (i.e., FSMC in [18, 19]) to describe the variation of the wireless channel condition. As shown in Figure 2, state transition only occurs between adjacent states. Since FSMC is derived by partitioning the received signal-to-noise ratio (SNR) into multiple intervals, there is a maximum feasible rate in each state with respect to the actual requirement of average Bit-Error Rate (BER).

3.2. SMDP Ingredients. In a general shared-channel wireless network, the scheduler should make scheduling decisions based on the knowledge of the system state, including the queueing delay of all packets, the channel conditions of all sessions, and the practical service rates of all sessions. Suppose that there are M sessions in the system, and each session has a queueing buffer that can store K packets at the most. (Without loss of generalization, we assume that all queues can store same number of packets. A more practical assumption may be that the buffer length of the mth session is K_m in packets.) At time t, the queueing delay of the kth packet in the mth session is denoted by $\tau_{m,k}(t)$. For the mth session, let $R_m(t)$ denote the maximum available channel rate at time t, $r_m(t)$ the practical average service rate at time t, and $\hat{r}_m$ the nominal service rate (negotiated when the session is setup). We now connect the packet scheduling problem with the main ingredients of the Semi-Markov Decision Process (SMDP) as follows.

3.2.1. Stage. The scheduling process is naturally divided into a series of stages by events $\omega_0, \omega_1, \omega_2, \ldots$, which represent packet arrivals and departures (we say that a packet "departs" if its service completes). Stages are indexed by integers $i = 1, 2, \ldots$. Let t_i denote the time that event ω_i happens. The time duration of the ith stage is represented by $\Delta t_i = t_i - t_{i-1}$.

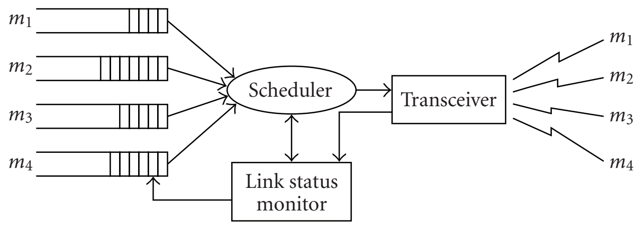


Figure 1: Scheduler model in a BWA network.

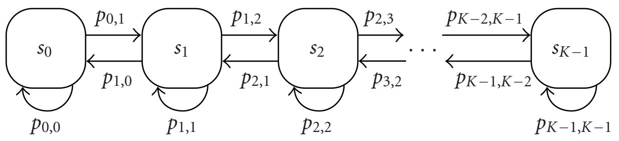


Figure 2: Multistate Markov channel model.

3.2.2. State. A state x consists of the queueing status of the whole scheduling system, including the current queueing delay of all packets, the wireless link conditions of all sessions, and the actual service rates of all sessions. In particular, the state at the beginning of stage i is defined by (for ease of presentation, we also call x_i the state of stage i in this paper)

$$x_i = \left(\tau^i_{m,k}, R^i_m, r^i_m \mid m \in \{1,2,\ldots,M\}, k \in \{1,2,\ldots,K\}\right), \tag{1}$$

where $\tau^i_{m,k} = \tau_{m,k}(t_{i-1})$, $R^i_m = R_m(t_{i-1})$, and $r^i_m = r_m(t_{i-1})$. For the computation of the actual service rate r^i_m, we improve the approach provided in [12]. We define a time window T_w which is sufficiently long. If session m is assigned for transmission, the actual service rate of session m at the end of the ith stage is updated by $r^{i+1}_m = (1-\Delta t_i/T_w)r^i_m + (\Delta t_i/T_w)R^i_m$; otherwise, $r^{i+1}_m = (1-\Delta t_i/T_w)r^i_m$. The set of all possible states is called *state space*, denoted by X, which is a multiple dimension space with huge size.

3.2.3. Decision. At stage i, with the occurrence of event ω_i, the scheduler should make a decision u_i according to the state x_i. The decision is defined by the session number of the next transmission, that is, $u_i = m$, $m \in \{1,2,\ldots,M\}$. The decision space is denoted by $U = \{1,2,\ldots,M\}$. At stage i, if event ω_i is packet departure, since the wireless channel is idle, the decision subset $U(x_i)$ contains all sessions that can transmit. (A session that "can transmit" is a session with nonempty queue and nonzero channel rate.) If event ω_i is packet arrival, in the case that the channel is busy (being used for transmission), $U(x_i)$ is a singleton; in the case that the channel is idle (as no session can transmit, e.g., an empty system), the decision subset $U(x_i)$ contains all sessions that can transmit.

3.2.4. Policy. Policy $\pi = (\mu_0, \mu_1, \mu_2, \ldots)$ is a sequence of decision functions, where decision function $\mu_i : X \mapsto U$ is a mapping from state space X to the decision space U. A policy is said to be *stationary* if for all i, $\mu_i \equiv \mu$, which means that the decision function does not change with stages. The space of stationary policy is denoted by Π_s. We only consider stationary policies as candidate policies in this paper, and without confusion, we simply use μ as the denotation of stationary policy. Note that, under policy μ, the decision at stage i is represented by $u_i = \mu(x_i)$.

3.2.5. Cost Function. As mentioned above, the definition of system cost of scheduling problem in BWA networks should take into account three objectives: (i) wireless bandwidth utilization maximization, (ii) QoS differentiation and guarantee, and (iii) fairness provisioning. Since cost function has major influence to the scheduler's performance, we place the construction of cost function in a single subsection which will be presented shortly. In the ith stage, the cost with state x_i and decision u_i is represented by $g(x_i, u_i)$. Under policy μ, the *average cost* of the system is provided by

$$v(\mu) = \lim_{N\to\infty} \frac{1}{t_N} \sum_{i=1}^{N} g(x_i, \mu(x_i)), \tag{2}$$

where N is the total number of stages. The target of the SMDP problem is to find the *optimal* policy which leads to the minimum average cost. A policy μ^* is said to be optimal if

$$v(\mu^*) \le v(\mu) \tag{3}$$

for arbitrary other policy μ. Usually, the average cost under the optimal policy is called optimal average cost, denoted by v^*.

3.2.6. Bellman Equation. Generally, the optimal policy μ^* of the SMDP problem could be obtained by solving the Bellman optimality equation. In the average cost SMDP problem, Bellman optimality equation takes the following form [20]:

$$\begin{gathered} v^* \mathbf{E}\{\Delta t \mid x\} + h^*(x) = \mathbf{E}\left\{ \min_{u\in U(x)} \left[g(x,u) + h^*(y) \right] \right\}, \\ h^*(\hat{x}) = 0. \end{gathered} \tag{4}$$

Here, $\mathbf{E}\{\cdot\}$ stands for expectation; Δt represents the duration of current stage; $U(x)$ denotes the decision set at current time; y is the state at next stage. The value v^* is the optimal average cost, and $h^*(\cdot)$ is called the optimal differential cost, which has the following interpretation: when we operate the system under an optimal policy, then $h^*(x) - h^*(x')$ equals to the expectation of the difference of the total cost (over the infinite horizon) for a system with initial state x, compared with a system with initial state x'. State $\hat{x}$ is a recurrent state with zero differential cost (e.g., the state in which all queues are empty).

Once the optimal differential cost function $h^*(\cdot)$ is available, the optimal scheduling policy can be obtained by

$$\mu^*(x) = \arg \max_{u \in U(x)} [g(x, u) + h^*(y)]. \tag{5}$$

As shown in (5), the optimal policy is composed by a series of optimal decisions at all stages, which maximizes the value of the righthand side of (5).

The desire for the exact solution of $h^*(\cdot)$ in (4) is unrealistic because of the huge size of state space and the overwhelming computational requirement. Hence, we employ the methodology of Reinforcement Learning (RL) [2] to produce a near optimal solution. The central idea is to approach $h^*(\cdot)$ by a parameterized approximating function $\widetilde{h}(\cdot, \theta)$. Online learning is carried out to continuously regulate the parameter vector θ and make $\widetilde{h}(\cdot, \theta)$ more and more close to $h^*(\cdot)$. The details of the RL solution are presented in the next section. Before that, we explain the construction of cost function in a separated subsection.

3.3. Construction of Cost Function. The definition of cost function directly determines the performance of the scheduling algorithm. In our problem, we restrict the cost function of interest to have the following important properties.

(i) *Simplicity*. A simple form of cost function can significantly reduce the computation complexity of the whole scheduling algorithm.

(ii) *Effectiveness*. Since the wireless packet scheduler is expected to concurrently address performance issues of QoS, bandwidth utility and fairness, the definition of cost function should take into account all these three aspects.

(iii) *Scalability*. In different practical services and applications, the requirements on every single performance objective may be quite different. Therefore, the structure of the cost function may have some tunable coefficients which allow the network supervisor to regulate according to actual situations.

Our discussion of the cost function starts from a *draining problem*, where given a number of packets in the system, and assuming no more packet arrivals, the scheduler should try to arrange the order of packet transmissions so that the total cost is minimized. Draining problem is indeed a simplified version of the original scheduling problem. By assuming that there is no new packet arrival, draining problem is much more analyzable than the original scheduling problem. Here, we consider a draining problem with the following setup. There are M sessions in the system. The initial queue length of session m is K_m in packets, $m \in \{1, 2, \ldots, M\}$. At the beginning, the queueing delay of all the packets is supposed to be zero. We separately define three different forms of cost function below. Each definition actually involves one performance objective.

3.3.1. Bandwidth Utilization. The first definition of cost function relates to the performance issue of bandwidth utilization. At the ith stage, let

$$g(x_i, u_i) = \sum_{m=1}^{M} \tau_{m,1}^i \mathbf{1}_{\{u_i = m\}}, \tag{6}$$

where $\tau_{m,1}^i$ is the head-of-line (HOL) delay of session m at the beginning of the ith stage; $\mathbf{1}_A$ is the indicator function, satisfying that $\mathbf{1}_A = 1$ when A is true, and $\mathbf{1}_A = 0$ when A is false. By this definition, the total cost is equal to sum of the delay of all packets. To minimize the total cost, the scheduler will always assign the session with the highest channel rate to transmit. Obviously, this scheduling policy will result in bandwidth utilization maximization.

3.3.2. QoS Differentiation. The second definition of cost function involves the performance issue of QoS differentiation. At the ith stage, let

$$g(x_i, u_i) = \sum_{m=1}^{M} W_m \tau_{m,1}^i \mathbf{1}_{\{u_i = m\}}, \tag{7}$$

where W_m is the weight with session m. The existence of weights makes the scheduling rule different from the one under definition (6). To be concrete, consider two sessions m_1 and m_2 with weights W_{m_1} and W_{m_2}, respectively, and $W_{m_1} > W_{m_2}$. Suppose at a certain stage that $R_{m_1} = R_{m_2}$, and both are the highest channel rates of all sessions. The scheduler will choose to assign m_1 for transmission first, because for a same period of waiting time, the cost of m_1 is larger than m_2. We can see that different weights give the session different priority levels. Following the definition of cost in (7), the scheduler can achieve QoS differentiation.

3.3.3. Fairness. The performance issue of fairness is integrated into the third definition of cost function. At the ith stage, let

$$g(x_i, u_i) = \sum_{m=1}^{M} \frac{\hat{r}_m}{r_m^i} \mathbf{1}_{\{u_i = m\}}, \tag{8}$$

where $\hat{r}_m$ and r_m^i are, respectively, the nominal and actual average service rate of session m. Under this definition, the scheduler is insensitive to the bandwidth utilization or QoS differentiation. What concerns the scheduler is whether every traffic gets its ordered service. The following theorem explains how the scheduler provides fair services for multiple traffic.

Theorem 1. *Considering a scheduling system with cost function of* (8) *and working under the Fluid limit model [21], if* *all sessions have the same channel rate R, the proportion of service that session m receives from the scheduler is given by* $\epsilon_m = \sqrt{\hat{r}_m}/\sum_{j=1}^{M}\sqrt{\hat{r}_j}$, *where the sum of service proportion is* $\sum_{m=1}^{M}\epsilon_m = 1$.

Proof. See the appendix. □

Theorem 1 indicates that under the definition of (8), each traffic flow will get its fair share of service being proportional to the square root of its nominal service rate. A more general form of cost function that relates to fairness issue is provided by extending the definition of (8) as follows:

$$g(x_i,u_i) = \sum_{m=1}^{M}\left(\frac{\hat{r}_m}{r_m^i}\right)^n \mathbf{1}_{\{u_i=m\}}, \quad n \geq 1. \tag{9}$$

It is easy to prove that the service proportion of session m under the definition of (9) has the form as $\epsilon_m = \hat{r}_m^{n/(n+1)}/\sum_{j=1}^{M}\hat{r}_j^{n/(n+1)}$.

Although none of the above three cost functions can simultaneously relate to all the three performance issues, they lead us to construct an appropriate one. Actually, by simply combining the definitions given by (6), (7), and (8), we obtain the cost function as follows:

$$g(x_i,u_i) = \sum_{m=1}^{M} W_m F_m(r_m)\tau_{m,1}^i \mathbf{1}_{\{u_{t_i}=m\}}, \tag{10}$$

where W_m's are constants called the priority factors, and F_m's are called the fairness factors, defined by

$$F_m(r_m) = \left(\frac{r_m}{\hat{r}_m}\right)^{-n_m}, \quad m \in \{1,2,\ldots,M\}. \tag{11}$$

Here $n_m \geq 1$ is a constant associated with session m that relates to fairness. The specific values of W_m's and n_m's are left to be determined according to the practical situations by the network supervisor. Regarding the form of the cost function in (10), it is worth to mention that a cost function of similar form is introduced in [11]. The authors assembled the factors of HOL delay, channel rate, and constant weight in a multiplication manner to produce an effective cost function. It is easy to see that the form of cost function definition by (10) has three important properties as mentioned at the beginning of this subsection. Firstly, it is simple and easy to compute in each stage. Secondly, as a combination of (6), (7), and (8), it effectively takes into account the three performance issues: bandwidth utilization, QoS guarantee, and fairness. Thirdly, the scalable coefficients W_m's and n_m's provide flexibility for the network supervisor to regulate the scheduler according to the practical requirements.

4. Reinforcement Learning Solution

4.1. Function Approximation Architecture. A common approximation architecture of RL is shown in Figure 3. The architecture consists of two components: the feature extractor and the function approximator. The feature extractor is

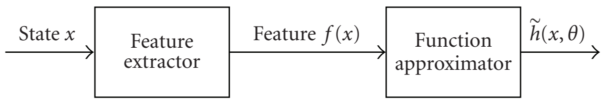


Figure 3: Architecture of RL system.

used to produce a feature vector $f(x)$, which is capable of capturing the most important aspects of state x. Features are usually handcraft, based on human intelligence or experience. Our choice of the feature vector will be presented shortly.

The function approximation architectures could be broadly categorized into two classes: the nonlinear and the linear structures. One typical nonlinear architecture is the multilayer perceptron or feedforward neural network with a single hidden layer (see, e.g., [22]). Under this architecture, the output approximating function $\hat{h}(\cdot,\theta)$ is the nonlinear combination of input feature vector $f(x)$ with internal tunable weight vector θ. This architecture is powerful because it can approximate arbitrary functions of $h^*(\cdot)$. The drawback, however, is that the dependence on θ is nonlinear, and tuning can be time-consuming and unreliable.

Alternatively, a linear feature-based approximation architecture [22] is much simpler and easier to implement, in which the output approximating function $\hat{h}(\cdot,\theta)$ is the linear combination of the input feature $f(x)$, given by

$$\tilde{h}(x,\theta) = \theta^T f(x), \tag{12}$$

where the superscript T represents transpose. In this paper, we choose linear approximation architecture in the RL framework.

Since the linear combination of feature vector should approximate the optimal differential average cost, the definition of feature vector explicitly relates to the definition of cost function. As (10) has shown, the cost at each stage is determined by the weighted queueing delay of the packet being assigned to transmit. Thus, the components of the feature vector should be defined with respect to all packets in the system, each component corresponding to one packet. Let l_m denote the packet size of session m. At stage i, the remaining waiting time for the kth packet in session m could be estimated by kl_m/r_m^i, where r_m^i is the current actual service rate of session m. Thus, the feature component with respect to the kth packet of session m is defined by

$$\xi_{m,k} = W_m F_m\left(r_m^i\right)\left(\tau_{m,k}^i + \frac{kl_m}{r_m^i}\right). \tag{13}$$

Here, the item $\tau_{m,k}^i + kl_m/r_m^i$ could be viewed as the estimation of the queueing delay for the kth packet in session m. Note that if r_m^i provides an accurate prediction of the service rate of session m, (13) coincides well with (10) and the loss of optimality of the linear architecture could be sufficiently small. Extending (12) yields

$$\tilde{h}(x,\theta) = \sum_{m=1}^{M}\sum_{k=1}^{K_m}\theta_m^k \xi_m^k, \tag{14}$$

where K_m is the number of packet in session m; θ_m^k is the parameter variable associated with feature ξ_m^k. The total number of tunable parameters in $\widehat{h}(\cdot,\theta)$ is equal to MK.

4.2. Online Parameter Tuning. After establishing the approximating architecture, we employ the technique of Temporal-Difference (TD) learning [2] to perform online parameter tuning. The algorithm starts with an arbitrary initial parameter vector θ_0 and improves the approximation as more and more state transitions are observed. At each stage, a greedy decision is made based on current x and θ. Specifically, the decision should be made by

$$u^* = \arg\min_{u\in U(x)} g(x,u) + \widetilde{h}(\widetilde{y},\theta), \tag{15}$$

where $\widetilde{y}$ is the estimation of state at the next stage. The computation of $\widetilde{y}$ is trivial if we simply assume that no packet arrives during the current stage. More concretely, let K_m denote the number of packets in session m. At the beginning of a stage, given the state

$$x = (\tau_{m,k}, R_m, r_m \mid m \in \{1,2,\dots,M\},\ k \in \{1,2,\dots,K\}), \tag{16}$$

if no more packet arrives before the current transmission completes, then the state at the end of current stage (i.e., the beginning of next stage) is estimated by

$$\widetilde{y} = \left(\tau'_{m,k}, R'_m, r'_m \mid m \in \{1,2,\dots,M\},\ k \in \{1,2,\dots,K\}\right), \tag{17}$$

where for $m \neq u$, $\tau'_{m,k} = \tau_{m,k} + l_u/R_m$, $k = 1,\dots,K_m$; for $m = u$, $\tau'_{m,k} = \tau_{m,k+1} + l_u/R_m$, $k = 1,\dots,K_{m-1}$.

In the process of parameter tuning, the update of parameter vector θ and scalar $\widetilde{v}$ is carried out stage by stage, according to

$$\begin{aligned}\theta_i &= \theta_{i-1} + \gamma_i d_i \nabla_\theta \widetilde{h}(x_{i-1},\theta_{i-1}),\\ \widetilde{v}_i &= \widetilde{v}_{i-1} + \eta_i[g(x_{i-1},u_{i-1}) - \widetilde{v}_{i-1}\Delta t_i],\end{aligned} \tag{18}$$

where γ_i and η_i are small step size parameters, and d_i is called temporal difference. Generally, γ_i and η_i should satisfy the following conditions:

$$\begin{aligned}&\sum_{i=0}^{\infty}\gamma_i = \infty, \qquad \sum_{i=0}^{\infty}\gamma_i^2 < \infty,\\ &\sum_{i=0}^{\infty}\eta_i = \infty, \qquad \sum_{i=0}^{\infty}\eta_i^2 < \infty.\end{aligned} \tag{19}$$

In (18), the temporal difference d_i is found as

$$d_i = [g(x_{i-1},u_{i-1})] - \left[\widetilde{v}_{i-1}\Delta t_i + \widetilde{h}(x_i,\theta_{i-1}) - \widetilde{h}(x_{i-1},\theta_{i-1})\right]. \tag{20}$$

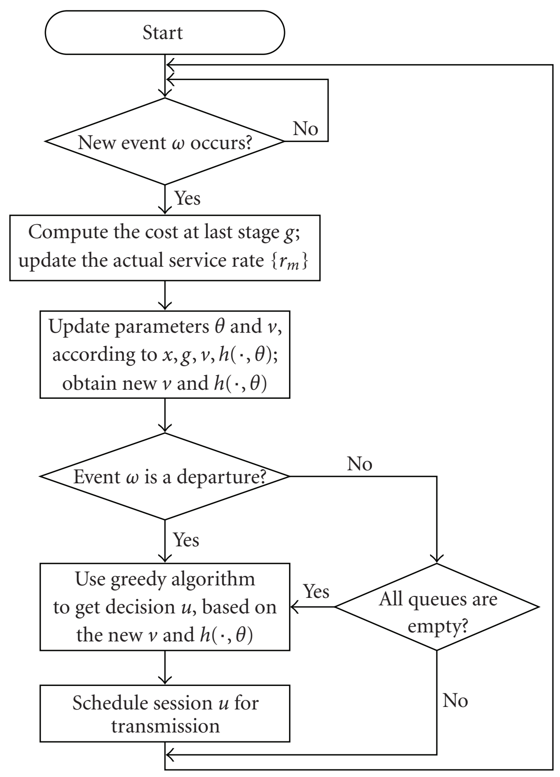

Start
New event ω occurs?
No
Yes
Compute the cost at last stage g; update the actual service rate $\{r_m\}$
Update parameters θ and v, according to $x, g, v, h(\cdot,\theta)$; obtain new v and $h(\cdot,\theta)$
Event ω is a departure?
No
Yes
Use greedy algorithm to get decision u, based on the new v and $h(\cdot,\theta)$
Yes
All queues are empty?
No
Schedule session u for transmission

Figure 4: Flow Chart of RLS.

Note that we use the linear approximation structure as (12), which leads to $\nabla_\theta \widetilde{h}(x,\theta) = f(x)$ in (18). We can see the benefit of using the linear feature-based approximation architecture which significantly reduced the complexity of gradient computation in (18). Figure 4 presents the flow chart of the proposed Reinforcement Learning-based Scheduling (RLS) algorithm, which summarizes the main operations of the algorithm.

5. Performance Evaluation

In this section, we evaluate the performance of RLS by network simulation. Two simulation experiments are carried out. The first experiment is to investigate the convergence performance of the parameter tuning procedure. The second one is to compare the performance of RLS with two existing packet scheduling algorithms: CSDP-WRR [17] and CIF-Q [9].

5.1. Simulation Setup. We consider three classes of traffic for typical BWA networks, including audio, video, and data traffic flows. As listed in Table 1, there are five sessions. All traffic flows are generated based on the models presented in [23]. Note that the traffic models are especially proposed for one of the BWA standards IEEE 802.16. In Table 1, the link conditions of audio, video, and data-1 are supposed to be stably full-rate, while link conditions of data-2 and data-3 are assumed to suffer drastic variation. More specifically,

Table 1: Properties of the 5 sessions in the simulation.

Session number (i)	1	2	3	4	5
Traffic type	Audio	Video	Data-1	Data-2	Data-3
Source model	IDP	2IRP	4IPP	4IPP	4IPP
Packet size	66 B	188 B	192 B	192 B	192 B
Mean rate	22.4 kbps	0.19 Mbps	0.8 Mbps	0.8 Mbps	0.8 Mbps
Peak rate	64 kbps	0.4 Mbps	1.8 Mbps	1.8 Mbps	1.8 Mbps
Link variation	None	None	None	Pattern-1	Pattern-2

Table 2: Channel state transition probabilities and steady-state probabilities ($i \in \{2, 3, 4, 5\}$).

Parameter	$p_{1,1}$	$p_{i,i-1}$	$p_{i,i+1}$	$p_{6,6}$	$\Pi = (\pi_1, \pi_1, \pi_2, \pi_4, \pi_5, \pi_6)$
Error-free	1	—	—	—	$(1, 0, 0, 0, 0, 0)$
Error pattern-1	0.92	0.40	0.40	0.50	$(0.52, 0.10, 0.10, 0.10, 0.10, 0.08)$
Error pattern-2	0.70	0.30	0.30	0.70	$(0.17, 0.17, 0.17, 0.17, 0.17, 0.15)$

link variability of data-2 and data-3 sessions is characterized by a six-state Markov channel model [18, 19]. Transmission rates of link states $s_1 \sim s_6$ are, respectively, 100, 80, 60, 40, 20, 0 percents of the full channel rate (2 Mbps). The steady-state probability vector for error pattern-1 is $\Pi_1 = (0.52, 0.10, 0.10, 0.10, 0.10, 0.08)$, while the one for pattern-2 is $\Pi_2 = (0.17, 0.17, 0.17, 0.17, 0.17, 0.15)$. The channel state transition probabilities of the data sessions are listed in Table 2.

Generally, audio and video traffic has higher priority than data traffic, due to the requirement of realtime transmission. Thus, we set the priority factors of the audio and video traffic flows as 1, that is, $W_{1,2} = 1$, and the priority factors of data traffic flows as 10^{-2}, that is, $W_{\{3,4,5\}} = 10^{-2}$. In the aspect of providing fair service, data traffic is usually more sensitive than audio and video traffic. This is because data traffic flows, such as FTP and e-mail, are generally served in the mode of best effort (BE). There is no guarantee how much service the system has to provide for these traffic flows. Without consideration of fair service, multiple data traffic flows in a same system may receive fairly different amounts of service, due to the different wireless channel conditions. Hence, we set larger fairness factors for the data traffic flows. The fairness factors of audio and video traffic are set as 1, that is, $n_{\{1,2\}} = 1$, and the fairness factors of data traffic are set as 3, that is, $n_{\{3,4,5\}} = 3$.

5.2. Convergence of Parameter Tuning. The first experiment investigates the convergence of RLS in the procedure of parameter tuning. Since the optimal average cost is the most important metric of the system performance, the variation of $\widetilde{\nu}$ reflects the convergence of the learning algorithm. We run the simulation and record the value of $\widetilde{\nu}$. The step size of the learning algorithm should be set before the simulation. As mentioned in Section 4, the setting of step size parameters γ_i and η_i should satisfy the conditions of (19). One efficient way to determine these parameters is to set the value of step size roughly inversely proportional to the stage number (or running time) [22]. Let $\gamma^i_{m,k}$ represent the step size with respect to feature component $\xi_{m,k}$ at stage i. We set

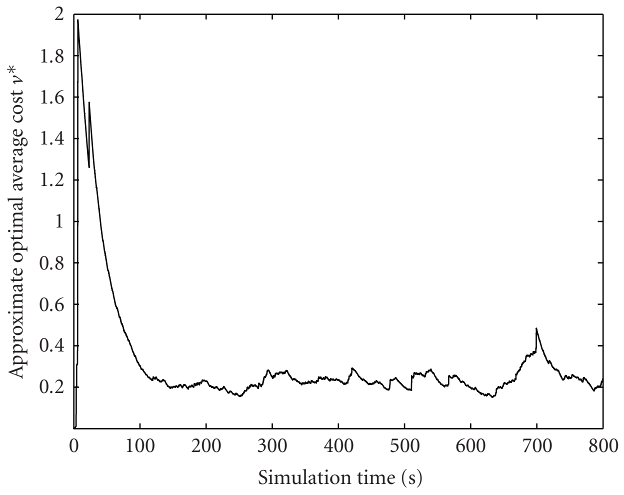


Figure 5: Curve of $\widetilde{\nu}$ under large step size parameters.

$$\gamma^i_{m,k} = \begin{cases} \dfrac{1}{1 + t_i/10} & \text{if } m \in \{1, 2\},\ k \in \{1, 2, \ldots, K\}, \\ \dfrac{10}{1 + t_i/10} & \text{if } m \in \{3, 4, 5\},\ k \in \{1, 2, \ldots, K\} \end{cases} \tag{21}$$

and the step size with the approximate optimal average cost $\widetilde{\nu}$ at stage i:

$$\eta_i = \frac{0.3}{1 + t_i/10}. \tag{22}$$

After running for 800 seconds, we record the variation of $\widetilde{\nu}$ and plot as curve in Figure 5.

It is observed that RLS converges very fast in the procedure of parameter tuning. The approximate optimal average cost $\widetilde{\nu}$ has reached a relatively stable value after 100 seconds. Although $\widetilde{\nu}$ keeps varying as time goes by, the

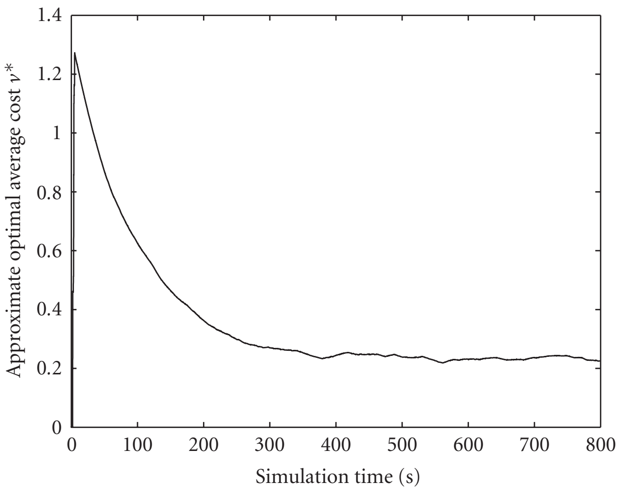


Figure 6: Curve of $\tilde{\nu}$ under small step size parameters.

fluctuation is still in an acceptable range. It is clear that the learning algorithm has converged. For comparison, we also choose another group of step size parameters. We set

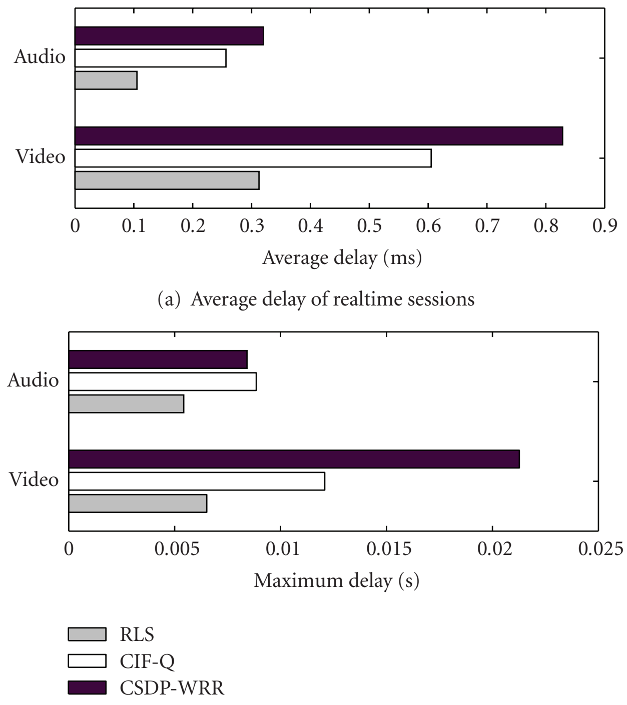
$$\gamma^i_{m,k} = \begin{cases} \dfrac{0.1}{1+t_i/10} & \text{if } m \in \{1,2\},\ k \in \{1,2,\ldots,K\}, \\ \dfrac{1}{1+t_i/10} & \text{if } m \in \{3,4,5\},\ k \in \{1,2,\ldots,K\}, \end{cases} \tag{23}$$

$$\eta_i = \frac{0.1}{1+t_i/10}, \tag{24}$$

where we can see that the step size parameters $\gamma^i_{m,k}$'s in (23) are 1/10 of the ones in (21), and parameters η_i's in (24) are 1/3 of the ones in (22). The second group of step size parameters is much smaller than the first group. The resulted curve of $\tilde{\nu}$ under the second group of step size parameters is plotted in Figure 6.

We observe in Figure 6 that RLS converges at a smaller speed than in the situation of Figure 5. It takes about 300 seconds for the scheduler to regulate the parameters and reach a stable approximate optimal average cost $\tilde{\nu}$. But the variation of the value of $\tilde{\nu}$ in Figure 6 is much more moderate than that in Figure 5. These results indicate that the smaller the step size parameters, the slower the convergence speed, but the stabler the system performance. The selection of step size parameters $\gamma^i_{m,k}$'s and η_i's depends on the actual network conditions. The scheduler should not only provide a sufficiently stable performance but also be fast reactive to follow the variability of outside environment.

5.3. Scheduling Performance. In the second experiment, we examine the performance of RLS in the aspects of bandwidth utilization, QoS guarantee, and fairness. We compare RLS with two existing scheduling algorithms CSDP-WRR [17] and CIF-Q [9]. CSDP-WRR is a combination of CSDP mechanism and Weighted Round Robin (WRR) scheduling algorithm, which has been demonstrated to be simple, robust, and effective [17]. The major feature of CIF-Q lies in the fairness it provides. CIF-Q integrates the well-known CIF-property, which takes into account both long-term and short-term fairness [9]. The simulation lasts for 1200 seconds. During the simulation time, channel errors occur only in the first 400 seconds. The remaining error-free time is to demonstrate the long-term fairness property of RLS. In the simulation, we keep track the delay, throughput, and service amount of the traffic flows. The results are reported in Figures 7, 8, and 9.



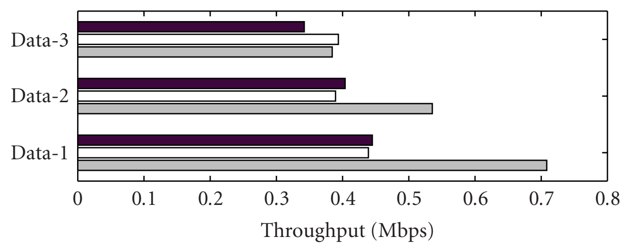
(a) Average delay of realtime sessions

(b) Maximum delay of realtime sessions

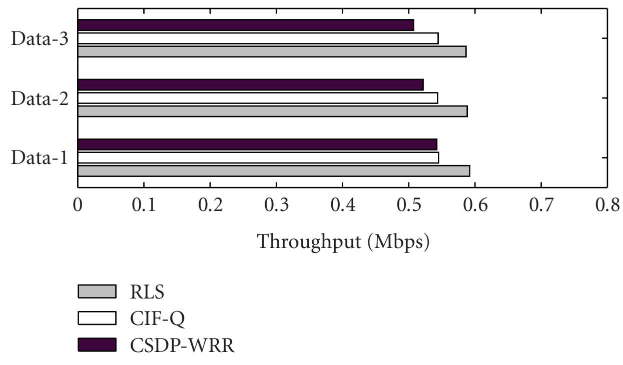
Figure 7: Comparison on delay of realtime sessions.

We know from Figures 7(a) and 7(b) that both the average and maximum delays of audio and video sessions in RLS are clearly smaller than in either of the other two algorithms. In other words, RLS is able to provide better QoS guarantee for realtime traffic flows. It should be noticed that the satisfying QoS guarantee for audio and video sessions in RLS is not obtained by sacrificing the throughput of the other sessions. This is validated by Figure 8.

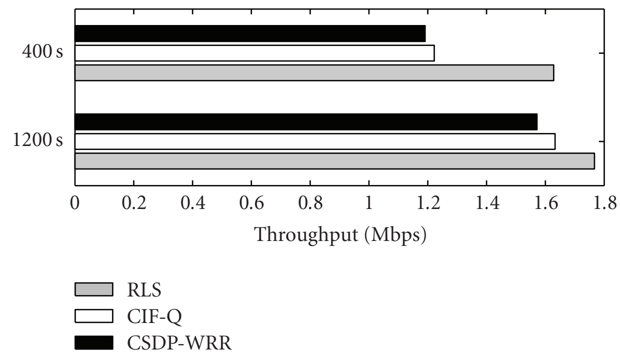
Figure 8 consists of three subfigures. Figures 8(a) and 8(b), respectively, report the throughput of data traffic flows in the first 400 seconds and the entire 1200 seconds. Figure 8(c) reports the total throughput of all the three data traffic flows. In Figure 8(a), compared to the other two algorithms, RLS significantly improves the throughput of data-1 and data-2, except that the throughput of data-3 in CIF-Q is a little more than that in RLS. But we know from Figure 8(c) that the total throughput of the three data traffic flows in RLS is much more than that in the other two algorithms. More specifically, in the first 400 seconds, the total throughput of data traffic is 1.62 Mbps, which is 37% higher than that in CSDP-WRR (1.19 Mbps) and 33% higher than that in CIF-Q (1.22 Mbps). In the entire 1200 seconds



(a) Throughput in the first 400 seconds



(b) Throughput in all 1200 seconds



(c) Total throughput of the three data sessions

Figure 8: Comparison on throughput of nonrealtime sessions.

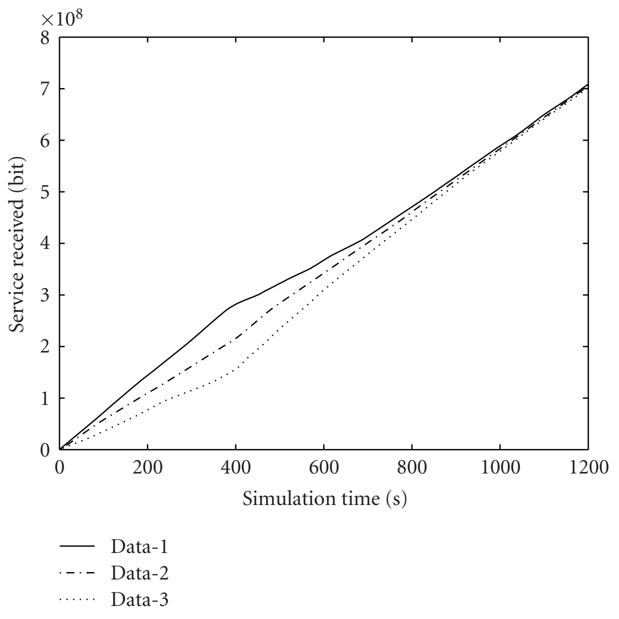


Figure 9: Service received by data sessions in NDPS.

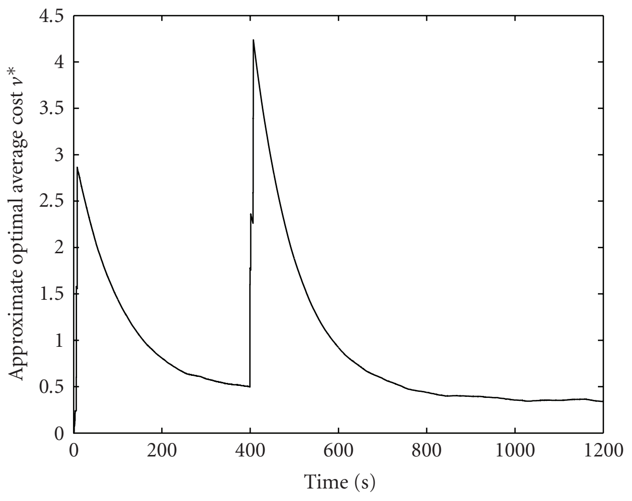


Figure 10: Curve of $\widetilde{\nu}$ in the second experiment.

of simulation, the overall throughput of data traffic in RLS is 1.77 Mbps, which is 13% higher than that in CSDP-WRR (1.57 Mbps) and 9% higher than that in CIF-Q (1.63 Mbps). Results of Figure 8 indicate that RLS has a considerable improvement in bandwidth utilization.

To demonstrate the short-term and long-term fairness properties of RLS, we record the service received by the data sessions over the whole simulation time in Figure 9. It is observed that the curves diverge moderately in the error-prone period and then converge gradually in the error-free period. We give interpretation as follows. To promise short-term fairness, RLS does not force the leading sessions to give up all their leads but just makes them degrade gracefully. So data-1 session receives relatively more service in the first 400 seconds. However, when the system becomes error-free, the lagging sessions will gradually get back their lags, and finally all data sessions obtain same throughput. This observation validates that RLS offers both short-term and long-term fairness guarantee in the sense that it provides short-term fairness for error-free sessions and long-term fairness for error-prone sessions. This exactly coincides with the spirit of CIF properties introduced in [9].

Finally, we plot in Figure 10 the curve of approximate optimal average cost $\widetilde{\nu}$ in the second experiment. Note that we use the group of small step size parameters defined by (23) and (24). As we set in the simulation, the network conditions are different in the first 400 seconds and the last 800 seconds (the channel models of data-2 and data-3 change at the 400th second). It is observed in Figure 10 that RLS, however, could still guarantee the convergence through online learning under the actual network conditions. Benefiting from its in-built learning ability, RLS could work adaptively according to the outside environment.

6. Conclusion

In this paper, we address the problem of packet scheduling in BWA networks. We cast the problem into the framework of semi-Markov Decision Process and solve it by the method of reinforcement learning. The proposed scheduling algorithm RLS simultaneously considers three performance issues in BWA networks: (i) QoS differentiation and guarantee, (ii) bandwidth utilization, and (iii) fair service. Being different from traditional scheduling algorithms, RLS is learning-based. It has the potential to learn from outside environment and adaptively regulate its scheduling policy. Simulation experiments are carried out to evaluate the performance of RLS. The numeric results indicate that RLS outperforms two existing scheduling algorithms in the sense that it provides better QoS for realtime traffic flows and meanwhile significantly improves the throughput of nonrealtime ones. Moreover, RLS achieves both long-term and short-term fairness.

Appendix

Proof of Theorem 1

Suppose that the total bandwidth of the system is γ. Let ϵ_m denote the proportion of bandwidth allocated by the scheduler to session m under the Fluid limit, that is, $R_m = \epsilon_m \gamma$. The scheduling problem could be formulated as a linear programming:

$$\begin{aligned} \min_{\epsilon_1,\epsilon_2,\ldots,\epsilon_M} \quad & \sum_{m=1}^{M} \frac{\hat{r}_m}{\epsilon_m \gamma}, \\ \text{s.t.} \quad & \sum_{m=1}^{M} \epsilon_m = 1. \end{aligned} \tag{A.1}$$

By introducing a positive Lagrangian multiplier λ, (A.1) can be rewritten as the following unconstrained optimization problem:

$$\min_{\epsilon_1,\epsilon_2,\ldots,\epsilon_M} L(\epsilon_1,\epsilon_2,\ldots,\epsilon_M) = \sum_{m=1}^{M} \frac{\hat{r}_m}{\epsilon_m \gamma} - \lambda\left(\sum_{m=1}^{M} \epsilon_m - 1\right). \tag{A.2}$$

For session m with nonzero service proportion ϵ_m, we let $\partial L(\epsilon_1,\epsilon_2,\ldots,\epsilon_M)/\partial \epsilon_m = 0$ and get

$$\epsilon_m = \sqrt{\frac{\hat{r}_m}{\lambda \gamma}}. \tag{A.3}$$

By substituting (A.3) into the constraint $\sum_{m=1}^{M} \epsilon_m = 1$, we have

$$\lambda = \frac{\left(\sum_{m=1}^{M} \sqrt{\hat{r}_m}\right)^2}{\gamma}. \tag{A.4}$$

The proportion of service for session m is finally given by

$$\epsilon_m = \frac{\sqrt{\hat{r}_m}}{\sum_{j=1}^{M} \sqrt{\hat{r}_j}}. \tag{A.5}$$

Acknowledgment

The work in this paper is supported by programs of NSFC under Grant nos. 60903170, U0835003, and U0635001.

References

[1] Y. Cao and O. K. Victor, "Scheduling algorithms in broadband wireless networks," *Proceedings of the IEEE*, vol. 89, no. 1, pp. 76–87, 2001.

[2] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, no. 1, pp. 9–44, 1988.

[3] L. Wischhof and J. W. Lockwood, "Packet scheduling for link-sharing and quality of service support in wireless local area," Tech. Rep. WUCS-01-35, November 2001.

[4] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, pp. 375–385, 1996.

[5] A. K. Parekh and R. G. Gallage, "A generalized processor sharing approach to fow control in integrated services networks: the single-node case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 344–357, 1993.

[6] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell System Technical Journal*, vol. 39, no. 9, pp. 1253–1265, 1960.

[7] E. O. Elliott, "Estimates of error rates for codes on burst-noise channels," *Bell System Technical Journal*, vol. 42, no. 9, pp. 1977–1997, 1963.

[8] S. Lu, V. Bharghavan, and R. Srikant, "Fair scheduling in wireless packet networks," *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 473–489, 1999.

[9] T. S. Eugen Ng, I. Stoica, and H. Zhang, "Packet fair queueing algorithms for wireless networks with location-dependent errors," in *Proceedings of the 17th Annual IEEE Conference on Computer Communications (INFOCOM '98)*, vol. 3, pp. 1103–1111, San Francisco, Calif, USA, March 1998.

[10] W. K. Wong, H. Zhu, and V. C. M. Leung, "Soft QoS provisioning using the token bank fair queuing scheduling algorithm," *IEEE Wireless Communications*, vol. 10, no. 3, pp. 8–16, 2003.

[11] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, P. Whiting, and R. Vijayakumar, "Providing quality of service over a shared wireless link," *IEEE Communications Magazine*, vol. 39, no. 2, pp. 150–153, 2001.

[12] A. Jalali, R. Padovani, and R. Pankaj, "Data throughput of CDMA-HDR a high efficiency-high data rate personal communication wireless system," in *Proceedings of the IEEE Vehicular Technology Conference (VTC '00)*, vol. 3, pp. 1854–1858, 2000.

[13] X. Liu, E. K. P. Chong, and N. B. Shroff, "Opportunistic transmission scheduling with resource-sharing constraints in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 10, pp. 2053–2064, 2001.

[14] S. Borst and P. Whiting, "Dynamic rate control algorithms for HDR throughput optimization," in *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '01)*, vol. 2, pp. 976–985, Anchorage, Alaska, USA, April 2001.

[15] D. Park, H. Seo, H. Kwon, and B. G. Lee, "Wireless packet scheduling based on the cumulative distribution function of user transmission rates," *IEEE Transactions on Communications*, vol. 53, no. 11, pp. 1919–1929, 2005.

[16] K. Wongthavarawat and A. Ganz, "Packet scheduling for QoS support in IEEE 802.16 broadband wireless access systems,"

International Journal of Communication Systems, vol. 16, no. 1, pp. 81–96, 2003.

[17] P. Bhagwat, P. Bhattacharya, A. Krishna, and S. K. Tripathi, "Enhancing throughput over wireless LANs using channel state dependent packet scheduling," in *Proceedings of the 15th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '96)*, vol. 3, pp. 1133–1140, March 1996.

[18] H. S. Wang and N. Moayeri, "Finite-state Markov channel—a useful model for radio communication channels," *IEEE Transactions on Vehicular Technology*, vol. 44, no. 1, pp. 163–171, 1995.

[19] Q. Zhang and S. A. Kassam, "Finite-state markov model for Rayleigh fading channels," *IEEE Transactions on Communications*, vol. 47, no. 11, pp. 1688–1692, 1999.

[20] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Vol. 1 and Vol. 2*, Athena Scientific, Belmont, Mass, USA, 1995.

[21] P. Liu, R. Berry, and M. L. Honig, "A fluid analysis of utility-based wireless scheduling policies," in *Proceedings of the 43rd IEEE Conference on Decision and Control (CDC '04)*, vol. 3, pp. 3283–3288, December 2004.

[22] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, Belmont, Mass, USA, 1996.

[23] C. R. Baugh and J. Huang, "Traffic Model for 802.16 TG3 MAC/PHY Simulations," IEEE 802.16 working group document, March 2001, http://wirelessman.org/tg3/contrib/802163c-01_30r1.pdf.