ELSEVIER Performance Evaluation 33 (1998) 89–112

PERFORMANCE EVALUATION An International Journal

Performance analysis of admission control algorithms based on reward optimization for real-time multimedia servers

Ing-Ray Chen [a,*], Tao-Hung Hsi [b]

[a] *Department of Computer Science, Virginia Polytechnic Institute and State University, Northern Virginia Center, 7054 Haycock Road, Falls Church, VA 22043, USA*

[b] *Institute of Information Engineering, National Cheng Kung University, Tainan, Taiwan, ROC*

Received 1 July 1997; received in revised form 7 December 1997; accepted 4 March 1998

Abstract

We propose and analyze a class of admission control algorithms based on reward optimization for multimedia servers designed to provide on-demand services to clients in an environment where workload characteristics of clients can change dynamically. These admission control algorithms are developed based on two strategic choices, namely, "deterministic" or "best-effort" regarding *quality of service* (QoS) *control*, and "priority-reservation" or "no priority-reservation" regarding *reservation control*. We first formulate the design of admission control algorithms for real-time multimedia servers as a reward optimization problem, with the "reward" referring to the value which the system receives after servicing prioritized clients based on the QoS requested and delivered. Then, we derive closed-form solutions for the reward rate expressions which the system can possibly obtain as a result of applying these admission control algorithms and validate the analytical results with a simulated VBR video server. A physical interpretation is given for the best combination of strategic choices under which the system can obtain the best reward as a function of workload and cost characteristics of the clients. We demonstrate that the reward value which the system receives under our proposed admission control algorithms at optimizing conditions can be much higher than those under traditional admission control algorithms which do not consider reward optimization. © 1998 Elsevier Science B.V. All rights reserved.

Keywords: Multimedia servers; Admission control; Quality of service (QoS); Capacity reservation; Client–server computing; Performance analysis

1. Introduction

On-demand multimedia servers including KTV servers and video-on-demand servers are frequently designed to service a large number of clients simultaneously [11,14]. Once a client is admitted, the service of media data normally must be provided continuously to the client based on the quality of service specified unless a re-negotiation takes place [18]. Due to this continuity requirement, a portion of the server capacity

* Corresponding author. Tel.: +1 703 538 8376; fax: +1 703 538 8348; e-mail: irchen@cs.vt.edu.

will necessarily be reserved for the client until the client departs. A central design issue of such multimedia servers thus is how to effectively allocate the server capacity to clients so that for a given hardware configuration more clients can be admitted without violating the QoS requirements of all the admitted clients. The issue is particularly interesting when the server is overloaded because new arriving clients might have to be rejected. In this case, a design issue is how to minimize the loss to the system, or, better yet, how to maximize the *pay-off* or *reward* to the system. This paper intends to formulate the design of admission control algorithms as a reward optimization problem.

Over the past few years, various admission control algorithms have been proposed in the literature, generally adopting the design concept that all the admitted clients are to be serviced in round-robin rounds where, in each round, media blacks requested by all admitted clients are retrieved, buffered, and transmitted across the network (or displayed locally) in a pipelined, continuous fashion. Each client is allocated with a buffer which alternates between being filled by disk reads and being emptied into the network (or displayed locally) in each round [1,12,13,17]. For example, if a client requests a playback rate of 30 frames/s, then in each service round, the system should retrieve as many media blocks as necessary, corresponding to a minimum of 30 frames from the server disk into the buffer area of the client, in order to meet the QoS requirement of that client. Obviously, the maximum number of clients which can be admitted by the server depends on the underlying hardware structure and capacity of the server and the QoS requirements of the clients.

Rangan et al. [13] have developed a round robin admission control algorithm to guarantee that the underlying hardware structure and capacity can satisfy the QoS requirements of all of the admitted clients. They allow clients to have different QoS requirements in terms of playback rates, and thus in each round different clients may request different numbers of media blocks to be retrieved. They developed a disk scheduling algorithm to efficiently retrieve the requested media blocks so as to reduce the retrieval time per client and for the system to be able to accept more clients. Based on their algorithm, the maximum number of clients which can be admitted to the system without violating their QoS requirements can be determined at the run time. Rangan and Vin [12] performed a similar analysis for a file system for digital video and audio. The approach adopted by Rangan et al. [12,13] is strategic in nature, i.e., not allowing the QoS requirement of any admitted client to be violated at any time. Such an admission strategy results in "deterministic" services where the continuity requirements of all admitted clients are never violated for any brief moment during the entire service span. Consequently, a worst-case assumption regarding service times, i.e., the worst-case retrieval time, is used in deriving the maximum number of clients which the server can admit while satisfying their QoS requirements.

Since deterministic services may needlessly constrain the number of clients that can be admitted and make the server capacity severely under-utilized, Vin et al. [17] proposed that the system may be able to admit a mix of *intolerant* and *tolerant* clients, with intolerant clients still requesting deterministic services while tolerant clients request only "predictive" services. The predictive service discipline is another strategy regarding QoS control for which momentary violations of the continuity requirements can be tolerated, as long as the fraction of media data delivered on time is larger than a specified threshold value, say α, specified as part of a client's QoS requirement. For example, $\alpha = 97\%$ means that 97% of media data must be delivered on time. For deterministic services, it is 100%. Based on this concept, Vin et al. derived admission control criteria for deterministic services (based on worst-case service times), and also for predictive services based on average-case service times obtained from extrapolation data from past measurements. They also discovered by simulation that the maximum number of clients which can be admitted by the server with predictive services is greatly increased with the server utilization being greatly improved, when compared with deterministic services. It was demonstrated that these benefits are obtained without sacrificing the QoS requirements specified by tolerant clients requesting predictive services. Although the work by Vin et al. [17] demonstrates the benefits of predictive services over deterministic services, it is not clear how a client would declare itself as a tolerant client and specify the threshold value α in his/her QoS requirement, since in real-world applications it is unlikely that a client would be willing to lose any data unforcefully. Since the work by Rangan et al. [12,13] in admission control is strategic in nature regarding QoS control, they again assume that when the system is overloaded, newly arriving clients are simply rejected.

In this work, we consider "deterministic" or "predictive" (which we term "best-effort" in this paper, to make it clearer) as only part of possible strategic choices in designing admission control algorithms for a multimedia server. In our view, an admission control algorithm is composed of a set of strategic choices, with "deterministic" and "best-effort" being the possible strategic choices in the strategy group concerning *QoS control*, while "priority-reservation" and "no priority-reservation" being the possible strategic choices in another group concerning *reservation control*. One strategic choice is to be selected out of each strategy group to form an admission control algorithm. [1] The deterministic or best-effort strategy concerns the maximum number of clients the system can admit based on the capacity and structure of the underlying hardware without violating the QoS requirements of all admitted clients; on the other hand, the priority-reservation or no priority-reservation strategy concerns whether the system should separately reserve server capacity to clients with different priority levels (and thus "values"), especially when the system is overloaded. This list of strategic choices considered in this paper is by no means exhausted. For example, for the strategy group regarding QoS control, in additional to the deterministic and best-effort strategic choices considered above, we can also include *statistical QoS control strategies* into this group. A statistical QoS control strategy determines if a client can be admitted based on a statistical estimation of the probability that the total data requested by all the users exceeds the server capacity. A new client to the system is rejected if this probability at the arrival instant is found to be greater than a specified threshold probability value, say, 10^{-4}. Chang and Zakhor [1] have proposed a class of statistical admission control strategies for buffer management for variable bit rate (VBR) video servers and showed that statistical admission control is more effective than deterministic admission control for interactive video server systems. This paper does not consider statistical admission control strategies.

The purpose of reserving the server capacity discriminatingly for different prioritized clients is for system "reward optimization", considering that a high-priority user is associated with a higher "value" if it is served successfully and, correspondingly, a higher "penalty" if it is rejected or served unsatisfactorily (due to QoS deterioration). With this concept, we formulate the design of admission control algorithms as a reward optimization problem, with the "design space" consisting of all possible combinations of strategic choices and with the goal being to optimize the system value.

Specifically, we consider that every client can be assigned with a reward indicating its value to the system (e.g., monetary value) when the client is successfully serviced, and conversely a penalty or, more generally,

[1] Another strategy group not considered in this paper consists of "local admission only" and "remote admission" strategic choices regarding *global QoS control*. "Local admission only" means that the server admits a client only locally and rejects the client if it cannot accommodate the client without forwarding the client to another replicated multimedia server, while "remote admission" means that the server forwards the client as often as necessary either because the client cannot be accepted locally or just for load balancing reasons. This strategy group offers the possibility of load balancing for a set of replicated multimedia servers providing the same service. This strategy group is not treated here since we do not deal with replicated servers in this paper.

a penalty function indicating the negative value (e.g., loss of profit) imparted to the system when the client is rejected or is not satisfied with the QoS delivered. These positive/negative reward functions reflect the benefit/loss to the server system. The problem we are interested in solving is determining the best admission control algorithm, according to which the system can receive the maximum value. In this paper, we consider the case in which the server services a mix of high-priority and low-priority clients, each characterized by its own arrival/departure rates as well as its reward/penalty value functions.

When we choose a strategy regarding QoS control, i.e., deterministic or best-effort, we can determine the maximum number of client requests that the system can service without overloading, based on previous theoretical results [13,17]. In addition, if we choose a "priority-reservation" strategy regarding reservation control, we can divide this maximum number of clients into several "priority threshold values," with a threshold value referring to the amount of server capacity reserved for the high-priority clients, for the low-priority clients, or both. We will show in this paper that there exists a set of optimizing "priority threshold values" under which the server can obtain the maximum reward. With the concept of reservation control for prioritized clients, we vitalize the role of admission control algorithms from a passive role, i.e., preventing system overload, to an active role, i.e., maximizing the system value while still satisfying the QoS requirements of all the admitted clients. With this formulation, the system value measure becomes a unifying performance metric in assessing the performance of admission control algorithms formed by various combinations of strategic choices. In this paper, we consider four possible algorithms since there are four possible combinations (i.e., 2 times 2) out of "deterministic" or "best-effort" regarding QoS control, and "priority reservation" or "no priority reservation" regarding reservation control.

The rest of the paper is organized as follows. In Section 2 we state the system assumptions and notation that are used in the paper. Section 3 develops analytical models based on queueing theory, for analyzing a class of "threshold-based" reservation control strategies. Based on the analysis result, we derive analytical expressions for the system reward rate (i.e. value/time) obtainable by the system for each of the four admission control algorithms considered in the paper. Section 4 validates the analytical results with a simulated variable bit rate (VBR) video server based on discrete-event simulations. A physical interpretation of the analytical and simulation results is given and the conditions under which one algorithm can provide a better system value than the others as a function of client workload characteristics are discussed. Finally, Section 5 summarizes the paper and outlines some future research areas.

2. Assumptions and system model

2.1. Client workload characteristics

We assume that our multimedia server system accepts clients with two different priority levels. The inter-arrival times of high-priority and low-priority clients are exponentially distributed with average times of $1/\lambda_h$ and $1/\lambda_l$, respectively. When a client arrives, the server determines whether to accept the client based on its admission control algorithm. Once a client is admitted and thus a reservation is made to the client, the client is assured of the reserved capacity until it departs. The inter-departure times of high-priority and low-priority clients are assumed to be exponentially distributed with an average time of $1/\mu$. While in general, different clients may have different QoS requirements regarding their playback rate, frame rate, display resolution, etc., for simplicity we assume that all clients demand the same QoS requirement. The last two assumptions are justified for video-on-demand (VOD) servers.

2.2. Cost model: Reward/penalty characteristics of clients

The value of a client to the server system is characterized by the client's reward/penalty cost model. We assume that when the server successfully services a high-priority client without violating its QoS requirement, the system receives a reward value of v_h. On the other hand, if a high-priority client is rejected upon arrival, we assume that the system loses a value of q_h immediately. Moreover, for a high-priority client having been admitted into the system, if the client is dissatisfied with the QoS delivered, then the system receives only a portion of the reward or even a penalty depending on the degree of discontent. Let α_h represent the percentage of multimedia data delivered to the client on time (thus representing the degree of QoS satisfaction) during the client's entire service span. Then, the reward value received by the server system when the client departs can be described by a reward function $v'_h(v_h, q_h, \alpha_h)$. For deterministic services, obviously $v'_h(v_h, q_h, \alpha_h) = v'_h(v_h, q_h, 1) = v_h$ since with deterministic services the QoS requirement of each admitted client is always satisfied and no client would be dissatisfied with the QoS delivered. For best-effort services, since the client's QoS requirement is not guaranteed all the time and α_h is not necessarily equal to one, a possible form of $v'_h(v_h, q_h, \alpha_h)$ is that it satisfies the following boundary conditions: if $\alpha_h = 1$, a positive value of v_h is returned and if $\alpha_h = 0$ then either zero is returned (in which case the reward value is down to zero but no penalty applies) or a negative value of $-q_h$ is returned (in which case a penalty applies). In the analysis which follows, we drop the argument part of v'_h and just use v'_h to denote this function; no assumption is made regarding its form. [2]

With the above cost model, a high priority client thus imparts a value of $v'_h(v_h, q_h, \alpha_h)$ when it is admitted into the server system and a negative value of $-q_h$ when it is rejected on admission; similarly, a low priority client imparts a value of $v'_l(v_l, q_l, \alpha_l)$ when it is admitted into the server system and a negative value of $-q_l$ when it is rejected on admission. When all clients are indistinguishable, these two sets of parameter values and functions are identical. While in reality each client might have its own set of reward/penalty parameters, for analytical tractability we assume that all high-priority clients have their unique set of (v_h, q_h, α_h) parameter values and reward function v'_h, and all low-priority clients also have their unique set of (v_l, q_l, α_l) parameter values and reward function v'_l. This assumption is reasonable for multimedia server systems which must provide distinct services to prioritized clients based on their values/penalties imparted to the system. We assume that the system designer can estimate the values of these two sets of parameters and reward functions to characterize the target client–server system.

2.3. Resource assumptions

We assume that the resources being controlled by a multimedia server include some processors and buffer, as well as a storage unit for storing multimedia data. While in general a storage unit may consist of tape, disk and expanded memory units (e.g., as in a video server [2]), we assume that a disk subsystem is

[2] In the simulation study to be introduced later in Section 4, we assume that $v'_h(v_h, q_h, \alpha_h)$ is of the exponential decay form, i.e.,

$$v'_h(v_h, q_h, \alpha_h) = \frac{v_h}{e^{\theta_h(1-\alpha_h)}}$$

which means that the reward value deteriorates from v_h to 0 exponentially as α_h goes from 1 to 0. Here θ_h is called the decay parameter which determines the degree of exponential deterioration of v'_h with respect to $1-\alpha_h$.

being used as the storage medium since it is a cost-effective solution with a reasonable latencies and data throughput rates [6].

Regardless of the admission control algorithm employed by the system, there is a maximum number of clients which can be admitted by the server. Although it is generally true that both the disk throughput and the buffer available can constrain the number of clients, we assume that the disk throughput is the limiting factor.

2.4. *Performance metric*

The performance metric being considered in their paper takes both rewards and penalties of the clients into consideration. It is called the system's total pay-off rate (or reward rate), defined as the average amount of reward received by the server per time unit. In other words, under a particular admission policy if the system on average services N_h high-priority clients and N_l low-priority clients per unit time while rejecting M_h high-priority clients and M_l low-priority clients per unit time, then the system total pay-off rate is

$$N_h v'_h + N_l v'_l - M_h q_h - M_l q_l.$$

Note that here we have used v'_h and v'_l instead of v_h and v_l for the reward functions of high-priority and low-priority, respectively, because we have to consider the degradation of the reward when a client does not satisfy the QoS delivered. The problem we are interested in solving is to identify the best admission control algorithm under which this performance metric is maximized.

Notation

λ_h	arrival rate of high-priority clients
λ_l	arrival rate of low-priority clients
μ	departure rate of clients
v_h	reward of a high-priority client if the client is serviced with a complete satisfaction with the QoS delivered
v_l	reward of a low-priority client if the client is serviced with a complete satisfaction with the QoS delivered
q_h	penalty of a high-priority client if the client is rejected on admission
q_l	penalty of a low-priority client if the client is rejected on admission
α_h	average degree of QoS satisfaction for high-priority clients
α_l	average degree of QoS satisfaction for low-priority clients
v'_h	short-hand notation for the reward function $v'_h(v_h, q_h, \alpha_h)$, which returns the actual reward value of a high-priority client
v'_l	short-hand notation for the reward function $v'_l(v_l, q_l, \alpha_l)$, which returns the actual reward value of a low-priority client
θ_h	decay parameter of a high-priority client for the case when the reward function of a high-priority client decays exponentially with the value of $1 - \alpha_h$, i.e., $v'_h(v_h, q_h, \alpha_h) = v_h \exp(-\theta_h(1 - \alpha_h))$
θ_l	decay parameter of a low-priority client
$\mathcal{PO}_x$	total system pay-off rate under a reservation control policy x
$\mathcal{PO}_x^y$	total system pay-off rate under a reservation control policy x and a QoS control policy y

n	maximum number of clients which the system can admit
n^{d}	maximum number of clients which the system can admit under deterministic QoS control
n^{b}	maximum number of clients which the system can admit under best-effort QoS control
n_{h}	number of slots reserved for high-priority clients, $0 \le n_{\mathrm{h}} \le n$
n_{l}	number of slots reserved for low-priority clients, $0 \le n_{\mathrm{l}} \le n$
n_{m}	number of slots reserved to service either types of clients, $n_{\mathrm{m}} = n - n_{\mathrm{h}} - n_{\mathrm{l}}$

3. Modeling and analyzing reservation and QoS control policies

In this section, we first propose the concept of "threshold-based reservation control" and derive closed-form expressions for the pay-off rate (or reward rate) which the system can obtain as a result of applying different threshold-based reservation control policies. Then, we derive reward rate expressions for four admission control algorithms formed by combining two threshold-based *reservation control* policies and two *QoS control* policies (i.e., deterministic and best-effort). Using the analytical expressions derived, we then analyze conditions under which one admission control algorithm can yield a higher reward rate than other ones.

Let n denote the maximum number of clients which can be admitted into the system based on a given QoS control strategy (i.e., deterministic or best-effort) as having been obtained analytically using the theoretical results derived in [12,13,17]. We model the capacity reservation mechanism of the system from an abstraction level as follows: We assume that each client, regardless of its priority type, imposes the same QoS requirement to the system and thus must reserve a fraction $1/n$ of the capacity. The system behaves as if it contains n capacity slots. When all slots are used-up, the server rejects a new arriving client so as to guarantee continuous services to all clients which have been admitted.

3.1. Threshold-based reservation control

3.1.1. Free-threshold

The simplest reservation control policy, termed "free-threshold," is to accept any new arriving client regardless its priority-type, as long as there is a slot to accommodate it (see Fig. 1). In essence, it is equivalent to the first-come-first-serve policy without applying any reward-based control. We will use this policy as the base policy against which other more sophisticated reservation control policies can be compared. Essentially, under this policy, there is no priority distinction among clients and the system behaves like a classic M/M/n/n system [7] with the arrival rate $\lambda = \lambda_{\mathrm{h}} + \lambda_{\mathrm{l}}$ and the departure rate $i\mu$ when there are i clients in the system. The loss rate of clients in this case is equal to

$$(\lambda_{\mathrm{h}} + \lambda_{\mathrm{l}}) \times \frac{(1/n!)[(\lambda_{\mathrm{h}} + \lambda_{\mathrm{l}})/\mu]^n}{1 + \sum_{j=1}^{n} (1/j!)[(\lambda_{\mathrm{h}} + \lambda_{\mathrm{l}})/\mu]^j},$$

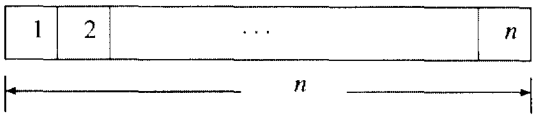


Fig. 1. Free threshold: n slots sharable.

where the first term is the collective arrival rate of high- and low-priority clients and the second term is the probability of all n slots having been occupied.

Using only one component in the state representation, the M/M/n/n model does not keep track of the number of high-priority or low-priority clients in each state. However, since with probability $\lambda_h/(\lambda_h+\lambda_l)$ a new client is a high-priority client and conversely with probability $\lambda_l/(\lambda_h+\lambda_l)$ it is a low-priority client, each state i (representing that there are i clients in the system in the steady state) can be associated with a pay-off reward rate of

$$i\mu \times \left(v_h' \times \frac{\lambda_h}{\lambda_h+\lambda_l} + v_l' \times \frac{\lambda_l}{\lambda_h+\lambda_l}\right).$$

On the other hand, in state n when all slots are used up, the penalty rate is given by

$$q_h \times \frac{\lambda_h}{\lambda_h+\lambda_l} + q_l \times \frac{\lambda_l}{\lambda_h+\lambda_l},$$

where we note that we have used v_h' and v_l' to denote the reward functions of a high-priority and a low-priority client, respectively, instead of just v_h and v_l. This is because under a QoS control policy (e.g., best-effort), the delivered QoS may not be perfect and hence v_h' (v_l') may be less than v_h (v_l correspondingly). Also note that n may vary depending on the QoS control policy employed. The value of n under best-effort services is normally much larger than that under deterministic services [13,17].

The system pay-off rate under the free-threshold policy, $\mathcal{P}O_{\text{free}}$, can be obtained by summing the pay-off reward rates weighted on their individual state probabilities, and subtracting the penalty rates due to rejection when all n slots are occupied, i.e.,

$$\begin{aligned}&\mathcal{P}O_{\text{free}}(n,\lambda_h,\lambda_l,\mu,v_h',v_l',q_h,q_l)\\&=\sum_{i=1}^{n} i\mu \times \left(v_h' \times \frac{\lambda_h}{\lambda_h+\lambda_l} + v_l' \times \frac{\lambda_l}{\lambda_h+\lambda_l}\right) \times \frac{(1/i!)[(\lambda_h+\lambda_l)/\mu]^i}{1+\sum_{j=1}^{n}(1/j!)[(\lambda_h+\lambda_l)/\mu]^j}\\&\quad-(q_h\lambda_h+q_l\lambda_l)\times\frac{(1/n!)[(\lambda_h+\lambda_l)/\mu]^n}{1+\sum_{j=1}^{n}(1/j!)[(\lambda_h+\lambda_l)/\mu]^j}.\end{aligned} \tag{1}$$

Note that here the total system pay-off rate $\mathcal{P}O_{\text{free}}$ is expressed as a function of n, λ_h, λ_l, μ, v_h', v_l', q_h, and q_l.

3.1.2. *Fixed-threshold*

Under the fixed-threshold reservation control policies, we allocate n_h slots, $n_h \le n$, to high-priority clients only, while the remaining slots $n_l = n - n_h$ slots are allocated to low-priority clients only (see Fig. 2). When all the slots allocated to high-priority (correspondingly low-priority) clients are exhausted, a new arriving high-priority (low-priority) client is rejected by the server even if there are still available slots

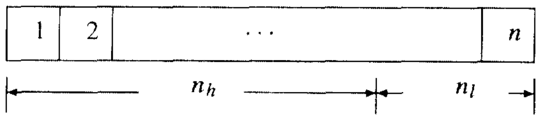


Fig. 2. Fixed threshold: n_h and n_l slots for high- and low-priority clients.

allocated to low-priority (correspondingly high-priority) clients. This situation is likely when we have a priori knowledge of the arrival rates of clients such that it is justified to reserve some capacity for high- or low- priority clients in order to maximize the system pay-off.

In this case, the server behaves like managing two separate, concurrent queues: one is an $M/M/n_\mathrm{h}/n_\mathrm{h}$ queue for high-priority clients only with the arrival rate equal to λ_h, service rate equal to $i\mu$ when there are i high-priority clients being serviced, and the slot size equal to n_h, while the other is an $M/M/n_\mathrm{l}/n_\mathrm{l}$ queue designated for low-priority clients only with the arrival rate equal to λ_l, service rate equal to $i\mu$ when there are i low-priority clients being serviced, and the slot size equal to n_l.

The loss rate of clients in this case is equal to the sum of that due to low-priority clients and that due to high-priority clients, i.e.,

$$\lambda_\mathrm{h} \times \frac{(1/n_\mathrm{h}!)(\lambda_\mathrm{h}/\mu)^{n_\mathrm{h}}}{1+\sum_{j=1}^{n_\mathrm{h}}(1/j!)(\lambda_\mathrm{h}/\mu)^j} + \lambda_\mathrm{l} \times \frac{(1/n_\mathrm{l}!)(\lambda_\mathrm{l}/\mu)^{n_\mathrm{l}}}{1+\sum_{j=1}^{n_\mathrm{l}}(1/j!)(\lambda_\mathrm{l}/\mu)^j}.$$

By associating a pay-off reward rate of $i\mu \times v_\mathrm{h}'$ (correspondingly $i\mu \times v_\mathrm{l}'$) for state i in the $M/M/n_\mathrm{h}/n_\mathrm{h}$ queue for high-priority (correspondingly in the $M/M/n_\mathrm{l}/n_\mathrm{l}$ queue for low-priority) clients and subtracting the penalty rate for the case when n_h (correspondingly n_l) slots are used-up for high-priority (low-priority) clients, we can compute the system pay-off rate as

$$\begin{aligned}
&\mathcal{P}O_{\mathrm{fixed}}(n_\mathrm{h}, n_\mathrm{l}, \lambda_\mathrm{h}, \lambda_\mathrm{l}, \mu, v_\mathrm{h}', v_\mathrm{l}', q_\mathrm{h}, q_\mathrm{l}) \\
&\quad = \sum_{i=1}^{n_\mathrm{h}} i\mu \times v_\mathrm{h}' \times \frac{(1/i!)(\lambda_\mathrm{h}/\mu)^i}{1+\sum_{j=1}^{n_\mathrm{h}}(1/j!)(\lambda_\mathrm{h}/\mu)^j} + \sum_{i=1}^{n_\mathrm{l}} i\mu \times v_\mathrm{l}' \times \frac{(1/i!)(\lambda_\mathrm{l}/\mu)^i}{1+\sum_{j=1}^{n_\mathrm{l}}(1/j!)(\lambda_\mathrm{l}/\mu)^j} \\
&\quad - \lambda_\mathrm{h} q_\mathrm{h} \times \frac{(1/n_\mathrm{h}!)(\lambda_\mathrm{h}/\mu)^{n_\mathrm{h}}}{1+\sum_{j=1}^{n_\mathrm{h}}(1/j!)(\lambda_\mathrm{h}/\mu)^j} - \lambda_\mathrm{l} q_\mathrm{l} \times \frac{(1/n_\mathrm{l}!)(\lambda_\mathrm{l}/\mu)^{n_\mathrm{l}}}{1+\sum_{j=1}^{n_\mathrm{l}}(1/j!)(\lambda_\mathrm{l}/\mu)^j} \qquad (2)
\end{aligned}$$

3.1.3. *Dynamic-threshold*

Under the dynamic-threshold reservation control policy, the n slots are divided into three parts: n_h, n_l and n_m, with n_h specifically allocated to high-priority clients, n_l allocated to low-priority clients while the remaining n_m slots sharable to both types of clients (see Fig. 3). When a high-priority (correspondingly low-priority) client arrives, if the number of slots having been occupied by high-priority (low-priority) clients is less than $n - n_\mathrm{l}$ (correspondingly $n - n_\mathrm{h}$), then the client is accepted; otherwise, it is rejected. This policy encompasses the previous two reservation control policies: in the case when $n_\mathrm{m} = 0$, this policy degenerates to the fixed-threshold reservation control policy, while in the case when $n_\mathrm{h} = n_\mathrm{l} = 0$, it degenerates to the free-threshold policy. It also covers the interesting case of $n_\mathrm{l} = 0$ wherein high-priority clients can use the n_m slots open to both high- and low-priority clients (as long as there is a space), but no low-priority clients can use any of the n_h slots allocated to high-priority clients.

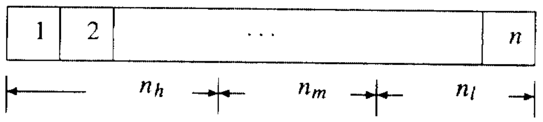


Fig. 3. Dynamic threshold: n_m slots sharable by high- and low-priority clients.

The closed-form solution to the system pay-off rate under the dynamic reservation control policy can be obtained by considering a two-level hierarchical model. The high-level model is like the one for the free-threshold reservation control policy, i.e., an $M/M/n_{\mathrm{m}}/n_{\mathrm{m}}$ queue with the arrival rate equal to $\Lambda_{\mathrm{h}}+\Lambda_{\mathrm{l}}$, and the service rate equal to μ. Here, Λ_{h} and Λ_{l} denote the arrival rates of high-priority and low-priority clients when n_{h} and n_{l} slots have been occupied by high-priority and low-priority clients, respectively. They can be obtained from two low-level models, one for each type of clients, like the ones we have used for the fixed-threshold reservation control policy. Specifically,

$$\Lambda_{\mathrm{h}} = \lambda_{\mathrm{h}} \times \frac{(1/n_{\mathrm{h}}!)(\lambda_{\mathrm{h}}/\mu)^{n_{\mathrm{h}}}}{1+\sum_{j=1}^{n_{\mathrm{h}}}(1/j!)(\lambda_{\mathrm{h}}/\mu)^{j}}, \qquad \Lambda_{\mathrm{l}} = \lambda_{\mathrm{l}} \times \frac{(1/n_{\mathrm{l}}!)(\lambda_{\mathrm{l}}/\mu)^{n_{\mathrm{l}}}}{1+\sum_{j=1}^{n_{\mathrm{l}}}(1/j!)(\lambda_{\mathrm{l}}/\mu)^{j}}.$$

The system pay-off rate in this case is the sum of that due to the n_{h} and n_{l} slots assigned to high- and low-priority clients only, and that due to the sharable n_{m} slots, i.e.,

$$\begin{aligned} &\mathcal{P}O_{\mathrm{dynamic}}(n_{\mathrm{h}}, n_{\mathrm{m}}, n_{\mathrm{l}}, \lambda_{\mathrm{h}}, \lambda_{\mathrm{l}}, \mu, v'_{\mathrm{h}}, v'_{\mathrm{l}}, q_{\mathrm{h}}, q_{\mathrm{l}}) \\ &\quad = \sum_{i=1}^{n_{\mathrm{h}}} i\mu \times v'_{\mathrm{h}} \times \frac{(1/i!)(\lambda_{\mathrm{h}}/\mu)^{i}}{1+\sum_{j=1}^{n_{\mathrm{h}}}(1/j!)(\lambda_{\mathrm{h}}/\mu)^{j}} + \sum_{i=1}^{n_{\mathrm{l}}} i\mu \times v'_{\mathrm{l}} \times \frac{(1/i!)(\lambda_{\mathrm{l}}/\mu)^{i}}{1+\mathrm{e}\sum_{j=1}^{n_{\mathrm{l}}}(1/j!)(\lambda_{\mathrm{l}}/\mu)^{j}} \\ &\qquad + \mathcal{P}O_{\mathrm{free}}(n_{\mathrm{m}}, \Lambda_{\mathrm{h}}, \Lambda_{\mathrm{l}}, \mu, v'_{\mathrm{h}}, v'_{\mathrm{l}}, q_{\mathrm{h}}, q_{\mathrm{l}}), \end{aligned} \tag{3}$$

where the expression for $\mathcal{P}O_{\mathrm{free}}(\cdots)$ is given earlier in Eq. (1). We check the boundary conditions below. For the special case when $n_{\mathrm{h}} = n_{\mathrm{l}} = 0$, we have $\Lambda_{\mathrm{h}} = \lambda_{\mathrm{h}}$, $\Lambda_{\mathrm{l}} = \lambda_{\mathrm{l}}$, the first two terms being zeros, and therefore $\mathcal{P}O_{\mathrm{dynamic}}\ (0, n, 0, \ldots) = \mathcal{P}O_{\mathrm{free}}(n, \ldots)$ in which the dynamic reservation control policy degenerates to the free reservation control policy. For the special case when $n_{\mathrm{m}} = 0$, we have $\mathcal{P}O_{\mathrm{free}}(0, \ldots) = -q_{\mathrm{h}}\Lambda_{\mathrm{h}} - q_{\mathrm{l}}\Lambda_{\mathrm{l}}$, and therefore $\mathcal{P}O_{\mathrm{dynamic}}(n_{\mathrm{h}}, 0, n_{\mathrm{l}}, \ldots) = \mathcal{P}O_{\mathrm{fixed}}(n_{\mathrm{h}}, n_{\mathrm{l}}, \ldots)$ in which the dynamic reservation control policy degenerates to the fixed reservation control policy. Eq. (3) correctly satisfies these boundary conditions.

3.2. Admission control algorithms: Combining reservation and QoS control policies

In the last section, we showed that a system can obtain different pay-off rates when it adopts different threshold-based reservation control policies. In this section, we combine two reservation control policies with two QoS control policies to form four admission control algorithms, based on reward optimization. The two reservation control policies considered are "free-threshold" and "dynamic-threshold" introduced earlier, while the two QoS control policies are "deterministic" and "best-effort." Our goal is to derive analytical expressions for the reward rates of the system under these four admission control algorithms and experimentally verify the results using simulation (Section 4). Let n^{d} and n^{b} denote the maximum numbers which can be admitted by deterministic and best-effort QoS control policies, respectively. As stated earlier, these numbers can be obtained using the previous theoretical results published elsewhere (e.g., [12,17]). In Section 4, we will show how these numbers can be obtained for a VBR video server.

3.2.1. Algorithm DNP: Deterministic and no priority-reservation

Our first admission control algorithm is formed by combining"deterministic" for QoS control with "no priority-reservation" (and thus free-threshold) for reservation control. In this case v'_{h} is simply equal to v_{h} because the QoS delivered to the client is always perfect under deterministic QoS control. Therefore, the reward rate obtainable under this admission control algorithm, denoted by $\mathcal{PO}_{\text{no priority}}^{\text{deterministic}}$, following Eq. (1), is given by

$$\mathcal{PO}_{\text{no priority}}^{\text{deterministic}} = \sum_{i=1}^{n^{\mathrm{d}}} i\mu \times \left(v_{\mathrm{h}} \times \frac{\lambda_{\mathrm{h}}}{\lambda_{\mathrm{h}}+\lambda_{\mathrm{l}}} + v_{\mathrm{l}} \times \frac{\lambda_{\mathrm{l}}}{\lambda_{\mathrm{h}}+\lambda_{\mathrm{l}}}\right) \times \frac{(1/i!)[(\lambda_{\mathrm{h}}+\lambda_{\mathrm{l}})/\mu]^{i}}{1+\sum_{j=1}^{n^{\mathrm{d}}}(1/j!)[(\lambda_{\mathrm{h}}+\lambda_{\mathrm{l}})/\mu]^{j}}$$
$$- (q_{\mathrm{h}}\lambda_{\mathrm{h}} + q_{\mathrm{l}}\lambda_{\mathrm{l}}) \times \frac{(1/n^{\mathrm{d}}!)[(\lambda_{\mathrm{h}}+\lambda_{\mathrm{l}})/\mu]^{n^{\mathrm{d}}}}{1+\sum_{j=1}^{n^{\mathrm{d}}}(1/j!)[(\lambda_{\mathrm{h}}+\lambda_{\mathrm{l}})/\mu]^{j}}. \tag{4}$$

Note that algorithm DNP corresponds to a traditional "deterministic" admission control algorithm as described in [12,17].

3.2.2. *Algorithm BNP: Best-effort and no priority-reservation*

We first note that with best-effort QoS control, $v_{\mathrm{h}} \geq v'_{\mathrm{h}}$ and $v_{\mathrm{l}} \geq v'_{\mathrm{l}}$ since the QoS delivered to the client may not always be perfect. Following the last derivation, the reward rate obtainable under this admission control algorithm, denoted by $\mathcal{PO}_{\text{no priority}}^{\text{best effort}}$, is given by

$$\mathcal{PO}_{\text{no priority}}^{\text{best effort}} = \sum_{i=1}^{n^{\mathrm{b}}} i\mu \times \left(v'_{\mathrm{h}} \times \frac{\lambda_{\mathrm{h}}}{\lambda_{\mathrm{h}}+\lambda_{\mathrm{l}}} + v'_{\mathrm{l}} \times \frac{\lambda_{\mathrm{l}}}{\lambda_{\mathrm{h}}+\lambda_{\mathrm{l}}}\right) \times \frac{(1/i!)[(\lambda_{\mathrm{h}}+\lambda_{\mathrm{l}})/\mu]^{i}}{1+\sum_{j=1}^{n^{\mathrm{b}}}(1/j!)[(\lambda_{\mathrm{h}}+\lambda_{\mathrm{l}})/\mu]^{j}}$$
$$- (q_{\mathrm{h}}\lambda_{\mathrm{h}} + q_{\mathrm{l}}\lambda_{\mathrm{l}}) \times \frac{(1/n^{\mathrm{b}}!)[(\lambda_{\mathrm{h}}+\lambda_{\mathrm{l}})/\mu]^{n^{\mathrm{b}}}}{1+\sum_{j=1}^{n^{\mathrm{b}}}(1/j!)[(\lambda_{\mathrm{h}}+\lambda_{\mathrm{l}})/\mu]^{j}}. \tag{5}$$

Note that algorithm BNP corresponds to a traditional "predictive" admission control algorithm as described in [12,17].

3.2.3. *Algorithm DP: Deterministic and priority-reservation*

With priority-reservation control, we can use either fixed-threshold or dynamic-threshold for reservation control. Since fixed-threshold reservation control is a special case of dynamic-threshold reservation control by setting $n_{\mathrm{m}} = 0$, we consider the use of the dynamic-threshold reservation control policy. The reward rate obtainable under algorithm DP, denoted by $\mathcal{PO}_{\text{priority}}^{\text{deterministic}}$, following Eq. (3), is given by

$$\mathcal{PO}_{\text{priority}}^{\text{deterministic}} = \sum_{i=1}^{n_{\mathrm{h}}^{\mathrm{d}}} i\mu \times v_{\mathrm{h}} \times \frac{(1/i!)(\lambda_{\mathrm{h}}/\mu)^{i}}{1+\sum_{j=1}^{n_{\mathrm{h}}^{\mathrm{d}}}(1/j!)(\lambda_{\mathrm{h}}/\mu)^{j}} + \sum_{i=1}^{n_{\mathrm{l}}^{\mathrm{d}}} i\mu \times v_{\mathrm{l}} \times \frac{(1/i!)(\lambda_{\mathrm{l}}/\mu)^{i}}{1+\sum_{j=1}^{n_{\mathrm{l}}^{\mathrm{d}}}(1/j!)(\lambda_{\mathrm{l}}/\mu)^{j}}$$
$$+ \mathcal{PO}_{\text{free}}(n_{\mathrm{m}}^{\mathrm{d}}, A_{\mathrm{h}}^{\mathrm{d}}, A_{\mathrm{l}}^{\mathrm{d}}, \mu, v_{\mathrm{h}}, v_{\mathrm{l}}, q_{\mathrm{h}}, q_{\mathrm{l}}), \tag{6}$$

where we have used the superscript "d" for n_{h}, n_{m}, n_{l}, A_{h} and A_{l} to denote that these slots are generated under the deterministic QoS control policy. Also, with deterministic QoS control in effect, $v'_{\mathrm{h}} = v_{\mathrm{h}}$ and $v'_{\mathrm{l}} = v_{\mathrm{l}}$ in Eq. (6).

3.2.4. Algorithm BP: Best-effort and priority-reservation

The reward rate obtainable under this algorithm, denoted by $\mathcal{PO}_{\text{priority}}^{\text{best effort}}$, following the last derivation, is given by

$$\begin{aligned}\mathcal{PO}_{\text{priority}}^{\text{best effort}} &= \sum_{i=1}^{n_{\text{h}}^{\text{b}}} i\mu \times v_{\text{h}}' \times \frac{(1/i!)(\lambda_{\text{h}}/\mu)^i}{1+\sum_{j=1}^{n_{\text{h}}^{\text{b}}}(1/j!)(\lambda_{\text{h}}/\mu)^j}\\ &+ \sum_{i=1}^{n_{\text{l}}^{\text{b}}} i\mu \times v_{\text{l}}' \times \frac{(1/i!)(\lambda_{\text{l}}/\mu)^i}{1+\sum_{j=1}^{n_{\text{l}}^{\text{b}}}(1/j!)(\lambda_{\text{l}}/\mu)^j}\\ &+ \mathcal{PO}_{\text{free}}(n_{\text{m}}^{\text{b}}, A_{\text{h}}^{\text{b}}, A_{\text{l}}^{\text{b}}, \mu, v_{\text{h}}', v_{\text{l}}', q_{\text{h}}, q_{\text{l}}),\end{aligned} \tag{7}$$

where we note that the differences between Eqs. (6) and (7) (and also between Eqs. (4) and (5)) are in the number of clients which the system can admit (n^{d} vs. n^{b}) and in the reward functions for clients which have been admitted into the system (v_{h} and v_{l} vs. v_{h}' and v_{l}').

3.3. Numerical examples

In this section, we give some numerical examples to illustrate the utility of our analysis result. Consider a multimedia server system in which there exist two classes of priority clients imparting different values to the system. The high-priority clients are characterized by (λ_{h}, μ, v_{h}', q_{h}) while the low-priority clients are characterized by (λ_{l}, μ, v_{l}', q_{l}). Furthermore, let v_{h}' and v_{l}' be of the exponential decay form such that

$$v_{\text{h}}' = \frac{v_{\text{h}}}{e^{\theta_{\text{h}}(1-\alpha_{\text{h}})}} \tag{8}$$

and

$$v_{\text{l}}' = \frac{v_{\text{l}}}{e^{\theta_{\text{l}}(1-\alpha_{\text{l}})}}. \tag{9}$$

Recall that θ_{h} (θ_{l}) is the decay parameter and α_{h} (α_{l}) is the extent of QoS satisfaction for high-priority clients (low-priority clients correspondingly). Fig. 4 plots the reward ratio $v_{\text{h}}'/v_{\text{h}}$ as a function of θ and α. When $\alpha = 1$ (as is the case under deterministic QoS control), $v_{\text{h}}' = v_{\text{h}}$; however, when $\alpha < 1$ (as is the case under best-effort QoS control), this ratio decreases exponentially with $\theta(1-\alpha)$. Consequently, when θ is a small value, e.g., $\theta = 1$, the ratio drops only slowly as α drifts away from 1; however, when θ is a large value, e.g., $\theta \geq 15$, this ratio drops sharply as α deviates from 1. Thus, the magnitude of θ reflects the consequence of a client's discontent with the QoS delivered in terms of the value imparted to the system after the client is serviced. The value of α refers to the QoS perceived by the client and should be at least as high as 0.95 for a valid server design [17].

Consider a case reported in [17] that under a given hardware configuration, the maximum number of clients which can be served concurrently was found to be around $n^{\text{d}} = 16$ if *deterministic QoS control* is adopted and $n^{\text{b}} = 84$ if *best-effort QoS control* is adopted with a read-ahead buffer of 1 media block per client. In this case, Tables 1 and 2 show that there exist optimal threshold value sets (n_{h}^{d}, n_{m}^{d}, n_{l}^{d}) and (n_{h}^{b}, n_{m}^{b}, n_{l}^{b}) under which the system pay-off rate is maximized for algorithms DP and BP, respectively, when given a set of model parameter values on (λ_{h}, λ_{l}, μ, v_{h}, v_{l}, q_{h}, q_{l}, α_{h}, α_{l}, θ_{h}, θ_{l}). Tables 1 and 2 are

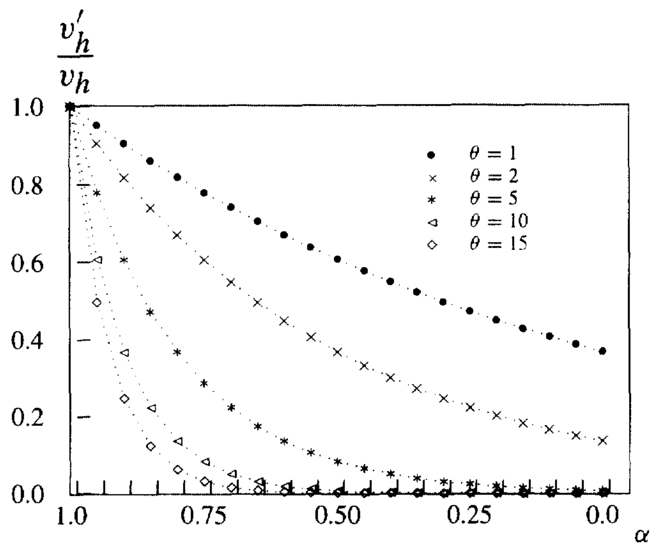


Fig. 4. Reward ratio ($v'_{\mathrm{h}}/v_{\mathrm{h}}$) received as a function of α and θ.

Table 1
Optimizing threshold value sets under algorithm DP ($n^{\mathrm{d}} = 16$)

$(\lambda_{\mathrm{h}}, \lambda_{\mathrm{l}}, \mu, v_{\mathrm{h}}, v_{\mathrm{l}}, q_{\mathrm{h}}, q_{\mathrm{l}})$	Optimal $(n^{\mathrm{d}}_{\mathrm{h}}, n^{\mathrm{d}}_{\mathrm{m}}, n^{\mathrm{d}}_{\mathrm{l}})$	Algorithm DP $\mathcal{PO}^{\text{deterministic}}_{\text{priority}}$	Algorithm DNP $\mathcal{PO}^{\text{deterministic}}_{\text{no priority}}$
(1, 10, 1, 2, 1, 2, 1)	(0,7,9)	12	11
(1, 10, 1, 5, 1, 2, 1)	(1,7,8)	15	14
(1, 10, 1, 10, 1, 2, 1)	(1,7,8)	19	18
(5, 10, 1, 2, 1, 2, 1)	(3,10,3)	15	14
(5, 10, 1, 5, 1, 2, 1)	(5,10,1)	28	27
(5, 10, 1, 10, 1, 2, 1)	(6,10,0)	52	48
(10, 10, 1, 2, 1, 2, 1)	(8,8,0)	14	12
(10, 10, 1, 5, 1, 2, 1)	(12,4,0)	41	33
(10, 10, 1, 10, 1, 2, 1)	(14,2,0)	88	69

Table 2
Optimizing threshold value sets under algorithm BP ($n^{\mathrm{b}} = 84$, $\theta_{\mathrm{h}} = 10$, $\theta_{\mathrm{l}} = 2$, $\alpha_{\mathrm{h}} = 0.995$, $\alpha_{\mathrm{l}} = 0.98$)

$(\lambda_{\mathrm{h}}, \lambda_{\mathrm{l}}, \mu, v_{\mathrm{h}}, v_{\mathrm{l}}, q_{\mathrm{h}}, q_{\mathrm{l}})$	Optimal $(n^{\mathrm{b}}_{\mathrm{h}}, n^{\mathrm{b}}_{\mathrm{m}}, n^{\mathrm{b}}_{\mathrm{l}})$	Algorithm BP $\mathcal{PO}^{\text{best effort}}_{\text{priority}}$	Algorithm BNP $\mathcal{PO}^{\text{best effort}}_{\text{no priority}}$
(10, 100, 1, 2, 1, 2, 1)	(7,60,17)	55	54
(10, 100, 1, 5, 1, 2, 1)	(10,68,6)	81	75
(10, 100, 1, 10. 1, 2, 1)	(12,72,0)	126	110
(50, 100, 1, 2, 1, 2, 1)	(50,34,0)	44	16
(50, 100, 1, 5, 1, 2, 1)	(56,28,0)	179	95
(50, 100, 1, 10, 1, 2, 1)	(60,24,0)	412	226
(100, 100, 1, 2, 1, 2, 1)	(84,0,0)	14	−56
(100, 100, 1, 5, 1, 2, 1)	(84,0,0)	244	63
(100, 100, 1, 10, 1, 2, 1)	(84,0,0)	627	261

generated by directly applying Eqs. (4)–(7). They show that algorithms DP and BP (shown in the third column) can provide a much higher value to the system than algorithms DNP and BNP (shown in the fourth column), respectively, as a result of applying the "dynamic-threshold" reservation control policy, under a variety of client workload and reward/penalty situations. In other words, the pay-off rate at the optimizing condition under algorithm DP (algorithm BP) is always better than that under under algorithm DNP (algorithm BNP correspondingly). Thus, by employing priority-reservation, i.e., reserving the server capacity separately for high- and low-priority clients, algorithms DP always performs better than algorithm DNP and algorithm BP always performs better than algorithm BNP.

Fig. 5 plots the performance gain of algorithm DP over algorithm DNP. (The trend is the same between algorithms BP and BNP which is not shown here.) The difference shown here is between the optimal pay-off rate under algorithm DP and that under algorithm DNP, with λ_h varying in the range of [10, 150] in increment of 10; λ_l varying in the range of [50, 150] in increment of 50; and $n = 100$. The trend exhibited in the graph when $v_h/v_l > q_h/q_l$ can be explained as follows. When the system load is light (corresponding to the left part of the graph) the effect of threshold-based priority-reservation control is not significant because the system can accommodate a new arriving client with a high probability, so that without priority-reservation control is just as good as with priority-reservation control. As the system load becomes moderate to heavy (corresponding to the middle part of the graph) the effect of threshold admission control becomes manifested because the server can effectively manage the server capacity (with thresholds) based on the knowledge regarding workload and reward/penalty characteristics of the clients so as to optimize the pay-off rate. Finally, when the load is very heavy (corresponding to the right part of the graph) the effect of priority-reservation control becomes less significant again because too many clients are rejected anyway even if the priority-reservation control is in effect.

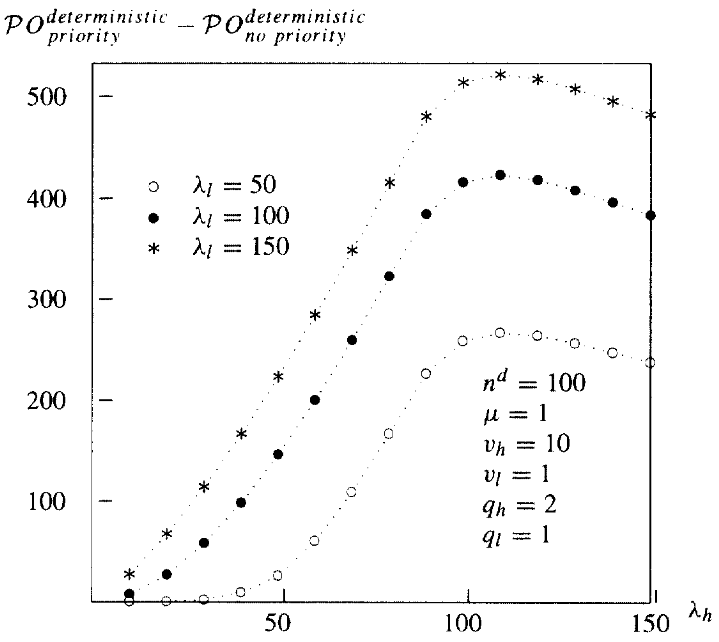


Fig. 5. Difference in reward rate between algorithms DP and DNP.

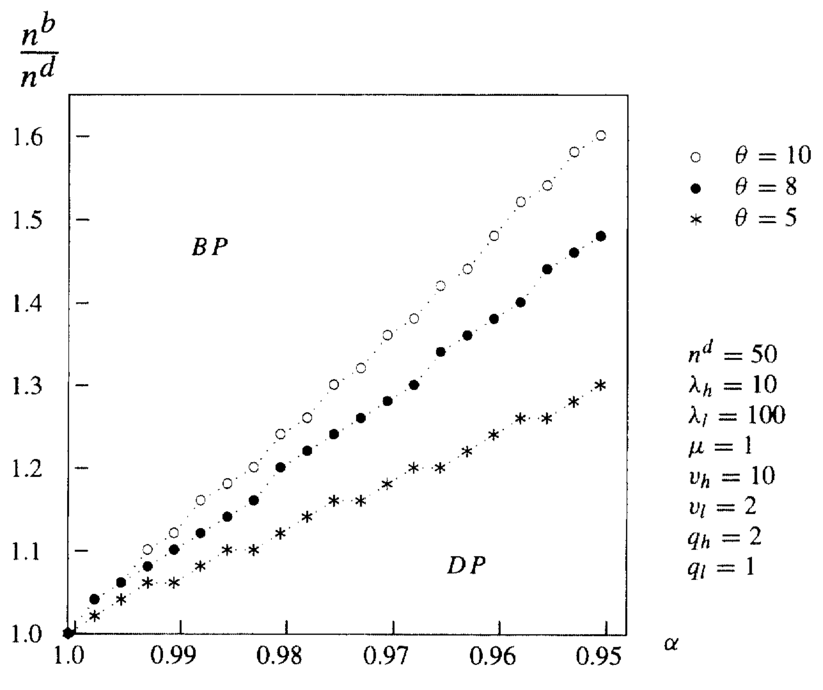


Fig. 6. Break-even curves between algorithms BP and DP.

Fig. 6 compares algorithms DP and BP and demonstrates that there exist conditions under which algorithm DP can perform better than algorithm BP, even though more clients can be admitted under algorithm BP than under algorithm DP. For comparison reasons, we let $\alpha_{\mathrm{h}} = \alpha_{\mathrm{l}} = \alpha$ and $\theta_{\mathrm{h}} = \theta_{\mathrm{l}} = \theta$. Here the x-axis is the ratio of n^{b} over n^{d} which is always at least 1 and the y-axis is α. There are several "break-even" curves in the figure, each for a distinct θ value. A point $(\alpha_x, (n^{\mathrm{b}}/n^{\mathrm{d}})_x)$ on a curve labeled with θ_x means that when $\alpha = \alpha_x$ and $\theta = \theta_x$, $n^{\mathrm{b}}/n^{\mathrm{d}}$ must be at least larger than $(n^{\mathrm{b}}/n^{\mathrm{d}})_x$ for algorithm BP to perform better than algorithm DP; otherwise, algorithm DP will perform better. For example, under the condition stated in Fig. 6, if $\theta = 8$ and $\alpha = 0.98$, then the number of clients admitted under algorithm BP must be at least 60 ($n^{\mathrm{b}} = n^{\mathrm{d}} \times 1.2$) to be able for algorithm BP to yield a better system reward than algorithm DP.

It should be noted that the trend exhibited in Fig. 6 is not universally true for all cases. It depends on the relative ratios of $v_{\mathrm{h}}/v_{\mathrm{l}}$ and $q_{\mathrm{h}}/q_{\mathrm{l}}$ and, in general, the workload and reward/penalty characteristics of the clients to the server system in question. Nevertheless, the equations derived earlier in this section can allow the designer to analytically identify the optimizing $(n_{\mathrm{h}}, n_{\mathrm{m}}, n_{\mathrm{l}})$ set when given the workload and reward/penalty characteristics of the clients, thus estimating how much benefit the system can gain (in terms of the system reward rate) as a result of applying an admission control algorithm based on reward optimization.

4. Simulation evaluation

In this section, we develop a simulated VBR MPEG-encoded video server to validate analytical results derived earlier and to further reveal the design trade-offs between algorithms DP and BP. Due to space limitation, here we only compare the reward rate metric. A more complete comparison of other performance metrics of multimedia servers including the maximum number of clients which the system can admit (i.e., throughput), the utilization of the server, the effective quality of service achieved, etc. can be found in [5]. The objective of the simulation study is to demonstrate the validity of the analytical result and to show its applicability.

4.1. *Simulation model and assumptions*

We first describe the model and assumptions used in implementing our simulated VBR MPEG-encoded video server.

4.1.1. *Data placement and retrieval*

We assume that the disk subsystem stores VBR MPEG-encoded video files. Each video file consists of a number of media blocks. The disk subsystem consists of a disk array of N_{disk} identical disks, each with N_{track} tracks. Each track contains N_{sector} sectors. A media block is a constant-data-length unit which is also a transfer unit between the disk subsystem and CPU. We assume that each media block is stripped across N_{disk} disks such that exactly N_{disk} sectors constitute a media block (with one sector on each disk). In other words, if D_{block} is the size in bytes of a media block, then $D_{\text{block}} = D_{\text{sector}} \times N_{\text{disk}}$ where D_{sector} is the size in bytes of a disk sector. When retrieving a media block from the disk subsystem, each of the N_{disk} disks simultaneously retrieves a sector of data, thus improving the transfer rate of the disk subsystem by a factor of N_{disk} when compared with a single disk system. We assume that successive blocks of a video file are stored on the disk subsystem randomly. When servicing clients' requests in each service round, the disk subsystem adopts the SCAN algorithm [15] to minimize the seek time by reordering accesses based on the locations of media blocks to be retrieved in that service round in the disk subsystem. Naturally, due to the VBR video data, the number of media blocks retrieved in each service round for a client may vary from one service round to another. To resolve this problem, the disk subsystem adopts a placement and retrieval scheme proposed by Chang and Zakhor [1]. Specifically, different numbers of constant-data-length media blocks can be retrieved in each service round, in accordance with a particular client's playback requirement demanded for the next service round. For example, if in a particular service round, the current buffer state of a particular client is 1.3 D_{block} of which 0.9 D_{block} is to be consumed for playback during that service round. Suppose that 2.8 D_{block} is needed for the next service round. Then, while 0.9 D_{block} is being consumed from the buffer, three media blocks will be read from the disk subsystem into that client's buffer so that at the end of the service round, the buffer state is 3.4 D_{block} which is large enough to cover the 2.8 D_{block} playback requirement for the next service round.

The information regarding the number of media blocks needed for a particular service round for a particular client can be obtained by inspecting the bit trace data of the video file being requested. We assume that such information is available to the server. For the simulation study, we obtain such information by using a VBR video bandwidth trace data set for a movie named "Star Wars" as disseminated by Bellcore [4]. The data set conforms to the MPEG-I standard; it consists of 174138 integers (in plain ASCII text), representing the number of bits per video frame. Only the original film frames are coded (i.e., 24 per second). The movie length is approximately 2 h.

We make use of this trace data set to simulate multiple movies being stored in the disk subsystem as follows: We fill in the disk subsystem one movie at a time until it is full. For each movie filled in, we randomly select a group of pictures [3] (GOP) in the first hour of the trace data set as the starting GOP for

[3] The sequence of MPEG I, P and B frames used is IBBPBBPBBPBB IBB..., so there are 12 frames in a group of pictures.

the movie, and then start filling in the disk subsystem one media block at a time randomly for this movie, until an hour of playing data are stored on the disk. As a result, each movie lasts for an hour and the server knows exactly where its successive, constant-data-length media blocks are placed (in a random manner) in the disk subsystem. The total number of movies stored on the disk array of the VBR video server turns out to be 55, consuming 99.5% of the disk space. An admitted client randomly selects one movie to view; each movie lasts exactly for an hour, thus making $\mu = 1/\mathrm{h}$ (see Table 4 later).

4.1.2. Buffer management

For each client having been admitted into the server system, we allocate a buffer space large enough to hold two rounds (i.e., two seconds) of "maximum" playing data. This is achieved by analyzing the trace data set to find out the "maximum" data size for one second of playing time. In the simulation, by using the trace data set obtained from Bellcore, we found this maximum size for one second of playing time to be 1.35 Mb. For example, if the size of a media block is 512 Kb then we allocate six media blocks of buffer space to each client upon a client's admission; this space is freed upon the client's departure. We manage this buffer space with a circular structure such that the tail part of the buffer is used for loading data from the disk subsystem, while the head part of the buffer is used for sending data previously retrieved over the network to the client. Since the disk subsystem stores VBR data (e.g., MPEG 1 encoding), the head and tail pointers of the buffer can change dynamically and must be managed by the simulation program for each client. For best-effort services, due to the constraint of the disk throughput, it is possible that the circular buffer allocated to a client may be underflowed at times, i.e., the data retrieved earlier in the buffer region between the head and tail pointers at times may be not rich enough to meet the playback requirement of the client. We will discuss how we handle this underflow situation later when we discuss admission control. For overflow situations, at no time will the tail pointer exceed the head pointer. If an overflow is about to occur during a round, then the server will simply stop retrieving data for that client during that round. Also, when the buffer of a client at any service round contains more than two rounds of playing data, then no retrieval will occur for that client during that service round. The saving in retrieval time may be useful in admitting more clients if the server is designed based on the best-effort QoS control policy. For each client admitted, the server maintains a data structure to record the current progress time of a movie he or she is watching so far, so by consulting the trace data set the server knows exactly how many media blocks should be retrieved during a service round to cover that client's playing requirement for the next service round. In the simulation, we let the length of a service round be 1 s.

4.1.3. Characteristics of prioritized clients

We assume that the workload as well as reward/penalty characteristics of prioritized clients are as described in Section 3.1. In addition, each client, regardless of its priority type, requests the same QoS regarding the playback rate (frame rate), display resolution, etc. for a video on demand (VOD) service. The penalty functions v_h' and v_l' for high-priority and low-priority clients, respectively, are defined as in Eqs. (8) and (9).

4.1.4. Interactive video functions

Since we allow the retrieval of more than one constant-data-length media block (if necessary) for any client in a service round, the simulated video server can satisfy interactive VOD functions such as fast forward or backward scan with a delay of one service round using the "skipping frames" technique. Since in the simulation we let the length of a service round be 1 s, so the jump delay is also 1 s.

4.1.5. *Admission control based on reward-optimization*

In the simulation model, we analyze all four admission control algorithms, namely, DP, DNP, BP and BNP, and study their design trade-offs.

We consider the following disk access time parameters (in milliseconds):

1. *Seek time:* We assume that the seek time of each hard disk in the disk array can be characterized by [17]:

$$t_{\mathrm{seek}}(\mathrm{track}_1, \mathrm{track}_2) = s_a + s_b * |\mathrm{track}_1 - \mathrm{track}_2|, \tag{10}$$

   which means that to go from one track (track_1) to another (track_2), it incurs an overhead time of starting the read–write head and an overhead time of moving the read–write head to the target track, with the latter quantity proportional to the distance between the source and target tracks; here s_a and s_b are constants.
2. *Rotational latency:* To read a media block from the disk subsystem, we assume that all disks of the disk array in the disk subsystem synchronize their read–write head movements, after which they simultaneously retrieve the media block. Therefore, when all the read–write heads all arrive at a track on which the target media block resides, all disks wait for a rotation time for the demanded sectors to come around before retrieving the target sectors simultaneously. We use t_{latency} to denote a full revolution time.
3. *Transfer time:* Since a constant-data-length media block is assumed to be stripped across all the disks in the disk array, the transfer time for a media block by the disk subsystem is equal to the transfer time of a sector by a single disk. Let R_{disk} be the read rate per disk. Then, the transfer time of one media block, denoted by t_{transfer}, is given by

$$t_{\mathrm{transfer}} = \frac{D_{\mathrm{block}}}{N_{\mathrm{disk}} R_{\mathrm{disk}}} = \frac{D_{\mathrm{sector}}}{R_{\mathrm{disk}}},$$

   where we note that the effective transfer rate of the disk subsystem is increased by a factor of N_{disk} because the size of the disk array in the disk subsystem is N_{disk}.

4.1.5.1. *Calculating n^{d} for algorithms DP and DNP.*

Recall that algorithm DP refers to "deterministic" for QoS control and "dynamic-threshold" for priority reservation control, while algorithm DNP refers to "deterministic" for QoS control and "free-threshold" for priority reservation control. Since these algorithms are deterministic in nature, we take a worst-case approach in calculating the theoretical bound on the maximum number of clients which the server system can admit without violating their QoS requirements. The maximum number of clients admitted under deterministic QoS control, i.e., n^{d}, can be determined a priori, viz,

$$n^{\mathrm{d}} = \left\lfloor \frac{T_{\mathrm{SR}} - s_{\mathrm{b}} N_{\mathrm{track}}}{\mathrm{NB}^{\mathrm{d}}_{\mathrm{pc}}(s_a + t_{\mathrm{rotation}} + t_{\mathrm{transfer}})} \right\rfloor, \tag{11}$$

where T_{SR} is the duration of a service round (i.e., 1 s or 1000 ms); $\mathrm{NB}^{\mathrm{d}}_{\mathrm{pc}}$ is the *worst-case* number of media blocks to be retrieved per client in one service round; and N_{track} is the number of tracks in a disk. The above expression results because we consider the worst case scenario in which all admitted clients must retrieve the largest playing data for the next second of playing time during the same service round. Note that in Eq. (11): (a) the seek time is separated into two terms because we assume that the SCAN algorithm is being used by the server; (b) in the worst case a full disk revolution for rotation time after a seek is needed before the demanded sector comes around; and (c) $\mathrm{NB}^{\mathrm{d}}_{\mathrm{pc}}$ is known a priori by consulting the VBR trace data set.

Table 3
Disk subsystem parameters

Symbol	Description	Value
N_{disk}	Number of disks in the disk array	16
N_{track}	Number of tracks in a disk	1024
N_{sector}	Number of sectors in a track	128
D_{block}	Number of bytes in a media block	64 KB
D_{sector}	Number of bytes in a sector	4 KB
s_a	Disk read–write head overhead time on every move	4 ms
s_b	Disk read–write head traversal time per track	0.02 ms
t_{latency}	Disk full rotation time	16.66 ms
R_{disk}	Disk read rate	5.4 MB/s
t_{transfer}	Disk transfer time for one media block of data	0.72 ms
T_{SR}	Service round time	1 s
$\mathrm{NB}_{\mathrm{pc}}^{\mathrm{d}}$	Worst-case number of media blocks to be read in one round per client	3
$\mathrm{NB}_{\mathrm{pc}}^{\mathrm{b}}$	Average-case number of media blocks to be read in one round per client	1 (initially)

In the simulation, the largest one-second playing data using the "Star Wars" VBR trace data set has the size of 1.35 Mb. Therefore, $\mathrm{NB}_{\mathrm{pc}}^{\mathrm{d}} = 3$ with the media block size being set to 512 Kb. Finally, we note that the same n^{d} value is obtained for both algorithms DP and DNP.

4.1.5.2. Admission control under algorithms DP and DNP. When we simulate algorithm DP or DNP, its n^{d} value is calculated based on Eq. (11), based on the set of disk subsystem parameter values specified (see Table 3). In particular, for algorithm DP, based on another given set of parameter values specifying the clients' workload and reward/penalty characteristics (see Table 4), a look-up is performed to map a particular n^{d} value to a set of optimal $(n_{\mathrm{h}}^{\mathrm{d}}, n_{\mathrm{m}}^{\mathrm{d}}, n_{\mathrm{l}}^{\mathrm{d}})$ values so as to control the number of high-priority and low-priority clients which can be admitted into the system (as discussed in Section 3). Such a look-up table is generated using the analytical results obtained in Section 3 and is encoded into the simulation program to map n^{d} into $(n_{\mathrm{h}}^{\mathrm{d}}, n_{\mathrm{m}}^{\mathrm{d}}, n_{\mathrm{l}}^{\mathrm{d}})$ automatically.

4.1.5.3. Calculating n^{b} under algorithms BP and BNP. While deterministic QoS control must consider the worst case scenarios so that no client will ever violate its QoS requirement, best-effort QoS control (as being used in algorithms BP and BNP) only tries to satisfy the QoS requirements of all admitted clients as

Table 4
Clients' workload and reward/penalty parameters

Symbol	Description	Value
λ_{h}	Arrival rate of high-priority clients	TBS
λ_{l}	Arrival rate of low-priority clients	TBS
μ	Departure rate of clients	1/h
v_{h}	Reward of a high-priority client	TBS
v_{l}	Reward of a low-priority client	TBS
q_{h}	Penalty of a high-priority client	TBS
q_{l}	Penalty of a low-priority client	TBS
θ_{h}	Decay parameter of a high-priority client	TBS
θ_{l}	Decay parameter of a high-priority client	TBS

TBS: To be specified in a simulation run.

much as possible. There are two changes made regarding admission control under best-effort QoS control when compared with deterministic QoS control:

1. instead of considering retrieving the worst-case number of blocks for each client during a service round, it considers retrieving only the *average-case* number of media blocks for each client;
2. instead of considering a full rotation time for each media block retrieved, it considers only one half of the full rotation time since on average one half disk revolution is needed for the read–write head to position on the desired media block.

Taking these changes, the maximum number of clients which can be admitted under either BP or BNP, denoted by n^{b}, is calculated as

$$n^{\mathrm{b}} = \left\lfloor \frac{T_{\mathrm{SR}} - s_b N_{\mathrm{track}}}{\mathrm{NB}^{\mathrm{b}}_{\mathrm{pc}}(s_a + 0.5t_{\mathrm{rotation}} + t_{\mathrm{transfer}})} \right\rfloor, \tag{12}$$

where $\mathrm{NB}^{\mathrm{b}}_{\mathrm{pc}}$ is the *average-case* number of media blocks to be retrieved per client in one service round, which is to be obtained statistically using the extrapolation data from past measurements. Note that $0.5t_{\mathrm{rotation}}$ has been used to account for the average rotation time. We again note that the same n^{b} value is obtained for both algorithms BP and BNP.

4.1.5.4. Admission control under algorithms BP and BNP. When simulating either algorithm BP or BNP, we initially estimate $\mathrm{NB}^{\mathrm{b}}_{\mathrm{pc}}$ to be 1 because the average data size for 1 s of playing time is 328 Kb by consulting the trace data set while the constant media block size is set to 512 Kb (see Table 3). We then re-estimate this value on a periodic basis using the statistical data collected from every past five-hours span. As a result, the number of clients which can be admitted under BP or BNP varies in every five hours until there is eventually a convergence. For all simulational experiments that we have run, we always observe a convergence. For algorithm BP, in particular, each n^{b} value obtained in each five-hour span is mapped into an optimal $(n^{\mathrm{d}}_{\mathrm{h}}, n^{\mathrm{d}}_{\mathrm{m}}, n^{\mathrm{d}}_{\mathrm{l}})$ using the analytical result obtained in Section 3 for a given set of parameter values characterizing clients' workload and reward/penalty conditions. Therefore, the numbers of high-priority and low-priority clients which can be admitted into the system also vary dynamically in every five hours until a steady-state is reached.

Another point that is worth mentioning is that BP or BNP, unlike its counterpart DP or DNP, can cause some clients to be starved whenever, in a service round, the accumulated number of blocks requested by all clients for the next service round exceeds that can be retrieved by the disk subsystem. One objective of our simulation is to obtain the parameter values of α_{h} and α_{l} to reflect the degradation of QoS due to the use of best-effort QoS control. We achieve this objective by monitoring the percentage of service rounds in which the playback requirement is satisfied for each admitted client. If in a service round a client's buffer state is rich enough to satisfy the playback requirement of that client for that service round, we increment its QoS counter by 1. Otherwise, we assume that a temporary pause has occurred to the client and the client's total viewing time is to be increased by one service round time. The value of α_{h} (or α_{l}) for a client is statistically calculated as the ratio of the QoS counter to the total viewing time of the client when that client departs. In case there exists a service round in which the accumulated requested number of blocks requested by all clients exceed that can be retrieved by the disk subsystem, we attempt to satisfy high-priority clients first. Specifically, we randomly select low-priority "victim" clients to cutoff their data retrievals until the total request rate matches with the disk retrieval rate. Randomly selecting low-priority victims has the effect of spreading out the starvation rounds to all low-priority clients, so that no single low-priority client will suffer too much QoS degradation. For most cases, we observe $\alpha_h = 1$, since high-priority clients are normally not selected as victims unless necessary. However, in some cases in which there are not too many low-priority clients being admitted into the system, it is still possible that $\alpha_h < 1$.

4.2. Simulation experiments and results

In the simulation model, we analyze all four admission control algorithms, namely, DP, DNP, BP and BNP, and study their design trade-offs. A simulation experiment run is set-up by assigning values to the model parameters. We separate our model parameters into two sets. The first set describes the characteristics of the disk subsystem, whereas the second set describes the clients' workload and reward/penalty characteristics. We fixed the values of the model parameters in the first set in all simulation runs, while allowing the values of the model parameters in the second set to be varied on a run by run basis to study the effect of clients' workload and reward/penalty characteristics on the design of admission control algorithms. These two sets of parameters are listed in Tables 3 and 4, respectively. By fixing the parameters values in the first set (in Table 3) for specifying the disk subsystem of the video server, we calculated $n^d = 15$ and $n^b = 75$ based on Eqs. (11) and (12), respectively. While the value of n^d remains fixed throughout a simulation run, the value of n^b varies in every 5 h, based on the statistical data collected during the past five hours until it eventually converges to 71. We run 120 h for each simulation run.

4.2.1. Comparison of theoretical and simulation pay-off rate values

Tables 5 and 6 compare the pay-off rate values obtained by simulation against those obtained analytically, for algorithms DP and BP, respectively. The pay-off rate values obtained by simulation are statistically collected on a client by client basis for 120 h of simulation time. Specifically, for each client admitted we keep track of its α value (i.e., the percentage of service rounds in which no starvation occurs) during its service period on a service-round by service-round basis. For algorithm DP, the α value for each client is always 1, whereas for algorithm BP it can be less than 1. When a client departs, we calculate its reward to the system based on the α value collected for that client according to Eq. (8) or (9), depending on whether the client is of low-priority or high-priority type. All such rewards for all clients are accumulated at the end of a simulation run, which is then divided by the total simulation time to obtain the average pay-off rate

Table 5
Comparison of analytical and simulation pay-off rate values for algorithm DP ($n^d = 15$)

$(\lambda_h, \lambda_l, \mu, v_h, v_l, q_h, q_l)$	Theoretical optimal (n_h^d, n_m^d, n_l^d)	Theoretical $\mathcal{PO}_{\text{priority}}^{\text{deterministic}}$	Simulation $\mathcal{PO}_{\text{priority}}^{\text{deterministic}}$
(1, 10, 1, 2, 1, 2, 1)	(0, 7, 8)	11	11
(1, 10, 1, 5, 1, 2, 1)	(1, 7, 7)	14	13
(1, 10, 1, 10, 1, 2, 1)	(1, 7, 7)	19	18
(5, 10, 1, 2, 1, 2, 1)	(3,10, 2)	13	13
(5, 10, 1, 5, 1, 2, 1)	(5,10, 0)	27	26
(5, 10, 1, 10, 1, 2, 1)	(7, 8, 0)	50	48
(10, 10, 1, 2, 1, 2, 1)	(8, 7, 0)	12	11
(10, 10, 1, 5, 1, 2, 1)	(12, 3, 0)	39	38
(10, 10, 1, 10, 1, 2, 1)	(15, 0, 0)	86	86
(10, 100, 1, 100, 10, 10, 1)	(15, 0, 0)	860	867
(10, 100, 1, 200, 20, 10, 1)	(15, 0, 0)	1823	1838

Table 6
Comparison of analytical and simulation pay-off rate values for algorithm BP ($n^b = 71, \theta_h = 10, \theta_l = 10$)

$(\lambda_h, \lambda_l, \mu, v_h, v_l, q_h, q_l)$	Theoretical optimal (n_h^b, n_m^b, n_l^b)	Theoretical $\mathcal{PO}^{\text{best effort}}_{\text{priority}}$	Simulation $\mathcal{PO}^{\text{best effort}}_{\text{priority}}$	α_h	α_l
(5,100,1,2,1,2,1)	(3,60, 8)	23	22	1	0.979
(5,100,1,5,1,2,1)	(5,66, 0)	37	37	1	0.980
(5,100,1,10,1,2,1)	(7,61, 3)	63	62	1	0.986
(10,100,1,100,10,10,1)	(15,56, 0)	1436	1392	1	0.991
(10,100,1,200,20,10,1)	(15,56, 0)	2929	2855	1	0.992
(50,100,1,2,1,2,1)	(53,18, 0)	18	18	1	0.973
(50,100,1,5,1,2,1)	(58,13, 0)	163	162	1	0.973
(50,100,1,10,1,2,1)	(62, 9, 0)	409	395	1	0.985
(100,100,1,2,1,2,1)	(71, 0, 0)	−52	−55	0.978	—
(100,100,1,5,1,2,1)	(71, 0, 0)	115	110	0.978	—
(100,100,1,10,1,2,1)	(71, 0, 0)	374	383	0.975	—

of the system as shown in Tables 5 and 6 (labeled with "Simulation $\mathcal{PO}$"). Tables 5 and 6 show that, as expected, the results obtained from simulation are quite consistent with those obtained analytically.

4.2.2. Cross-over points under which DP is better than BP

Fig. 7 displays a case which shows converged $\mathcal{PO}$ values for four admission control algorithms as a function of θ, with $\theta_h = \theta_l = \theta$ which varies in ratio of 2 from 1 to 1024, and $\lambda_h = 4/\text{h}$, $\lambda_l = 100/\text{h}$, $\mu = 1/\text{h}$, $v_h = 200$, $v_l = 20$, $q_h = 10$, $q_l = 1$. The values of α_h and α_l were statistically calculated from

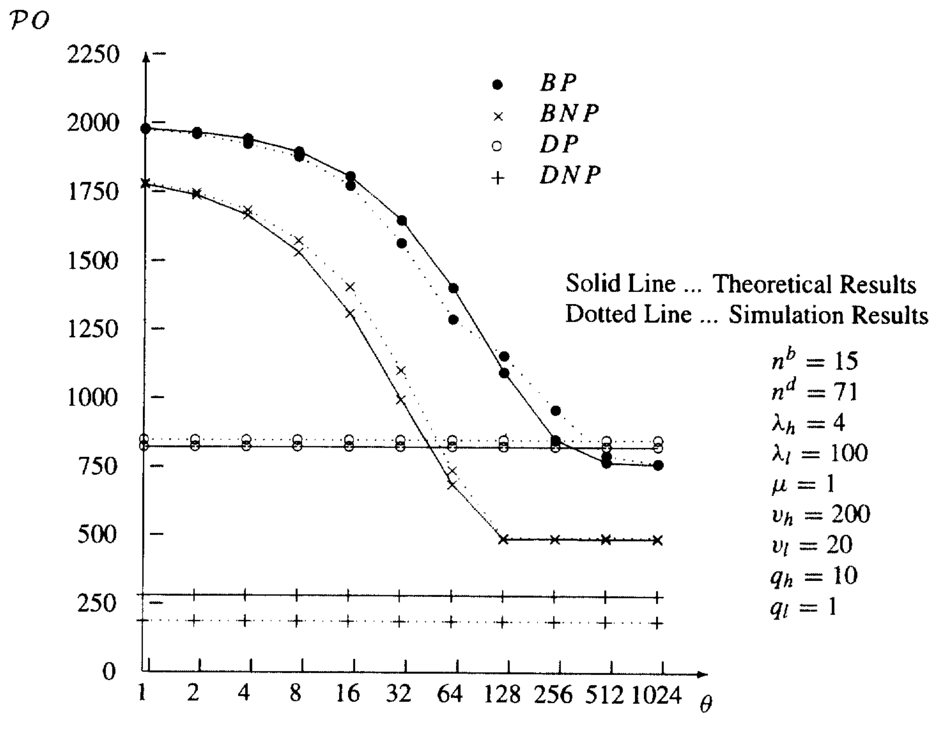


Fig. 7. Comparison of payoff rates among four algorithms as a function of θ.

the simulation data. Fig. 7 shows that DP becomes the best algorithm when θ is greater than a threshold value, i.e. 512 in this case. This represents the case when no client is willing to tolerate even a slight degree of QoS degradation, thus making any client's reward degrade very rapidly if its α value is not equal to 1. Fig. 7 thus unveils the cross-over conditions under which DP, although admitting a smaller number of clients (i.e. 15), can still perform better than algorithm BP which admits a larger number of clients (i.e. 71). Fig. 7 also shows that, as expected, algorithm DP always performs better than algorithm DNP and algorithm BP always performs better than algorithm BNP. We have observed such a trend in all experiments we have run.

Each data point shown in Fig. 7 represents a pay-off rate value which the system can obtain under a particular algorithm. Fig. 7 also demonstrates that the analytical pay-off rate values based on Eqs. (4)–(7) (shown in solid lines) correlate closely with simulation pay-off rate values (shown in dotted lines).

5. Summary

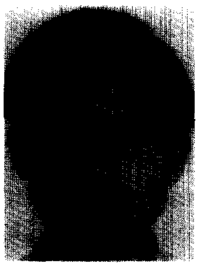
In this paper, we described a concept of designing admission control algorithms for multimedia servers based on reward optimization. We separated system parameters into two sets: one for describing the hardware characteristics of the server and one for describing the clients' workload and reward/penalty characteristics. We advocate that an admission control algorithm can be composed of a QoS control strategy and a priority-reservation strategy. We proposed and analyzed four admission control algorithms formed by combining two QoS control strategic choices and two priority-reservation strategic choices. We derived analytical solutions for the reward rate (in terms of the reward received by the system per unit time) the system can receive under each of these four algorithms, and then validated the analysed results and demonstrated the applicability of our approach by using a simulated VBR video server. Important conclusions drawn from our analysis include: (a) an algorithm that considers priority-reservation always performs better than one that does not make such consideration in terms of yielding a better system pay-off value; (b) for a wide range of parameter values we have analyzed, the best strategic combination is "best-effort" for QoS control and "priority" for reservation control; however, there exist conditions under which an algorithm that combines "deterministic" for QoS control and "priority" for reservation control can perform the best; and (c) the reward and penalty characteristics largely affect whether one admission control algorithm should be adopted over the others, especially for heavily-loaded systems since admitting more clients while delivering to them degraded QoS may not be beneficial to the system as opposed to admitting less clients while delivering 100% QoS to the clients.

The exact condition under which one admission control algorithm can best maximize the system total pay-off over the other ones depends on the hardware characteristics of the server, the relative ratios of v_h/v_l and q_h/q_l and, in particular, the workload and reward/penalty characteristics of the clients to the server in the target system. For example, if the penalty functions v'_h and v'_l are of a different form from Eqs. (8) and (9) (say, returning a negative value instead of zero when $\alpha = 0$), then we may find more cases in which algorithm DP performs better than algorithm BP. The analytical and simulational tools developed in this paper can be used by a system designer to evaluate these possible admission control algorithms based on the characteristics of the target system before they are actually implemented.

Some future research areas related to this paper include (a) considering the case when QoS negotiations or re-negotiations are possible; and (b) applying the reward optimization concept to the design of a distributed multimedia server which can be scalable to the clients' QoS requirements, i.e., performing a dynamic replication of the server or a load balancing policy with the goal of optimizing a specified, global reward metric.

References

[1] E. Chang, A. Zakhor, Admission control and data placement for VBR video servers, Int. Conf. on Image Processing, 1994, pp. 278–282.
[2] Y.N. Doğanta, A.N. Tantawi, Making a cost-effective video server, IEEE Multimedia Winter (1994) 22–31.
[3] K. Fujikawa et al., Application level QoS modeling for a distributed multimedia system, 1995 Pacific Workshop on Distributed Multimedia Systems, 1995, pp. 44–51.
[4] M.W. Garrett, A. Fernandez, Variable bit rate video bandwidth trace using MPEG code, Bellcore; host = ftp.bellcore.com; directory = pub/vbr.video.trace; file = MPEG.data.
[5] T.H. Hsi, Admission control algorithms based on reward optimization for real-time multimedia servers, MS Thesis, Institute of Information Engineering, National Cheng Kung University, June 1996.
[6] K. Keeton, Storage alternatives for video service, Proc. 13th IEEE Symp. on Mass Storage Systems, June 1994, pp. 100–105.
[7] L. Kleinrock, Queueing Systems, vol. 1: Theory, Wiley, New York, 1975.
[8] G.G. Mei, M.H. Lin, L. Hu, H. Chang, A real-time multimedia system for video applications, Proc. 26th IEEE Conf. on Signals, Systems and Computers, 1992, pp. 1031–1036.
[9] C.W. Mercer, S. Savage, H. Tokuda, Processor capacity reserves: Operating system support for multimedia applications, Proc. 1st IEEE Int. Conf. on Multimedia Computing and Systems, Boston, 1994, pp. 90–99.
[10] E. Oomoto, K. Tanaka, OVID: Design and implementation of a video-object database system, IEEE Trans. Know. Data Eng. 5 (4) (1993) 629–643.
[11] Y.J. Oyang, C.H. Wen, C.Y. Cheng, M.H. Lee, J.T. Li, A multimedia storage system for on-demand playback, IEEE Trans. Consumer Electron. 41 (1) (1995) 53–64.
[12] P.V. Rangan, H.M. Vin, Designing file systems for digital video and audio, Proc. 13th ACM Symp. on Operating Systems Principles, 1991.
[13] P.V. Rangan, H.M. Vin, S. Ramanathan, Designing an on-demand multimedia service, IEEE Comm. 30 (July 1992) 56–64.
[14] S. Ramanathan, P.V. Rangan, Architecture for personalized multimedia, IEEE Multimedia (Spring 1994) 37–46.
[15] A. Silberschatz, P.B Galvin, Operating System Concepts, 4th ed., Addison-Wesley, MA, 1994.
[16] A. Vina, J.L. Lerida, A. Molano, D. del Val, Real-time multimedia systems, Proc. 13th IEEE Symp. on Mass Storage Systems, 1994, pp. 77–83.
[17] H.M. Vin, A. Goyal, P. Goyal, Algorithms for designing multimedia servers, Comput. Comm. 18 (3) (1995) 192–203.
[18] A. Vogel, B. Kerherve, G.V. Bochmann, J. Gecsei, Distributed multimedia and QoS: A survey, IEEE Multimedia 2 (2) (1995) 10–18.



Ing-Ray Chen received his BS degree from the National Taiwan University, and the MS and PhD degrees in Computer Science from the University of Houston, University Park. He is currently an Associate Professor in the Department of Computer Science at Virginia Tech. His research interests are reliability and performance analysis, and real-time intelligent systems. Dr. Chen is a member of the IEEE/CS and ACM.

Tao-Hung Hsi received his BS and MS degrees in Computer Science from Soochow University in 1994 and National Cheng Kung University in 1996, respectively. His research interests are multimedia and mobile computing systems.