Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE @ DIRECT®

Performance Evaluation 52 (2003) 1–13

PERFORMANCE EVALUATION
An International Journal

www.elsevier.com/locate/peva

Performance evaluation of an admission control algorithm: dynamic threshold with negotiation[☆]

Sheng-Tzong Cheng[a,*], Chi-Ming Chen[a], Ing-Ray Chen[b]

[a] *Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, ROC*
[b] *Computer Science Department, Virginia Polytechnic Institute and State University, Falls Church, VA 22043, USA*

Received 9 December 1998; received in revised form 10 June 2002

Abstract

An admission control algorithm for a multimedia server is responsible for determining if a new request can be accepted without violating the QoS requirements of the existing requests in the system. Most admission control algorithms treat every request uniformly and hence optimize the system performance by maximizing the number of admitted and served requests. In practice, requests might have different levels of importance to the system. Requests offering high contribution or reward to the system performance deserve priority treatment. Failure of accepting a high-priority request would incur high penalty to the system.

A novel threshold-based admission control algorithm with negotiation for two priority classes of requests is proposed in our previous study. The server capacity is divided into three partitions based on the threshold values: one for each class of requests and one common pool shared by two classes of requests. Reward and penalty are adopted in the proposed system model. High-priority requests are associated with higher values of reward as well as penalty than low-priority ones. In this paper, given the characteristics of the system workload, the proposed analytical models aim to finds the best partitions, optimizing the system performance based on the objective function of the total reward minus the total penalty. The negotiation mechanism reduces the QoS requirements of several low-priority clients, by cutting out a small fraction of the assigned server capacity, to accept a new high-priority client and to achieve a higher net earning value. Stochastic Petri-Net model is used to find the optimal threshold values and two analytical methods are developed to find sub-optimal settings. The experiment results show that the sub-optimal solutions found by the proposed analytical methods are very close to optimal ones. The methods enable the algorithm to dynamically adjust the threshold values, based on the characteristics of the system workload, to achieve higher system performance.
© 2002 Elsevier Science B.V. All rights reserved.

Keywords: Admission control; Negotiation; Multimedia systems; Stochastic Petri-Net; Queuing theory

1. Introduction

Delivering multimedia streams with QoS requirements to viewers is one crucial issue in designing a multimedia system. The server of such a system requires admission control policy to guarantee the

[☆] This work is sponsored by the National Science Council, Taiwan, ROC, under the contract NSC 88-2213-E-006-057.
[*] Corresponding author. Tel.: +06-275-7575x62500; fax: +06-274-7076.
E-mail address: stcheng@mail.ncku.edu.tw (S.-T. Cheng).

delivery of on-demand multimedia streams. Upon the arrival of a new request, the server decides if the request can be admitted based on the availability of the server capacity. QoS guarantee of continuous multimedia stream delivery is met, once it is admitted. One mechanism to admission control is based on the *reservation scheme* [6,16]. The reservation scheme allocates a fraction of the server capacity (e.g. CPU time and network bandwidth) for a new request based on certain criteria. The reserved capacity is used to retrieve a specified number of disk blocks, to perform multimedia data processing, and to transmit data on the allocated channel. The allocated server capacity is reserved for the specific request until it leaves the system. A new request may be rejected if no available resource is left to serve the request. In such a case, the system incurs loss due to the rejected requests. As described above, an efficient admission control policy is essential to maximize the system performance and to reduce the loss rate.

In literature, various admission control algorithms have been proposed. The deterministic approach derives a formula of the maximum number of admitted requests under the worst-case load [1]. The requests are assured of their QoS requirements throughout their existence in the system. A system using such a deterministic approach might be under-utilized, since the admission control policy is based on the worst-case scenarios. The deterministic approach represents one extreme of the spectrum in the admission control algorithms, while the observation-based approach stands for the other extreme [5]. The latter approach is based on the prediction from the measurements of the resource usage status [2–5,7,10,12,13] and provides predictive service guarantee to clients, not absolute guarantee. The basic idea of such an observation-based algorithm is to aggressively accept a request as long as the acceptance of the request does not violate the service guarantee of the existent requests. The statistical approach [5] assumes that the average data access time does not change significantly, and it admits new clients as long as the server can meet the statistical estimation of the total data rate. In the paper [2], the proposed adaptive admission control algorithm admits new clients based on the extrapolation from the past measurements of the storage server performance.

The above research does not consider different priorities of client requests. Most research [11] tries to admit as many requests as possible without considering the importance of each request. We observe that, in some systems, clients might offer high value of reward and should be given to priority services. Similarly, the system pays high penalty if it rejects a high-priority request. Different priorities are associated with different values of reward and penalty. The admission control policy for such a system attempts to maximize the net earning (the total reward minus the total penalty) in order to optimize the system performance.

A class of threshold-based admission control algorithms, based on the above cost model, is proposed in our previous study [6]. The server capacity is partitioned into several partitions based on the threshold values: one for each class of requests and possibly one common pool shared by all classes. Requests of a specific priority are granted as long as the current load for the priority class is below the corresponding threshold. The server capacity from the common pool can only be used if the priority class requests have used up all the corresponding reserved partition of the server capacity. The admission control algorithm reaches an optimal objective value by dynamically adjusting the threshold values—server partitions, based on the characteristics of the system workload.

We further observe that the system could reach a higher objective value by lowering the service quality of admitted low-priority clients, so as to make room for new arrival of high-priority clients. Such an observation is exploiting the human perceptual tolerance [5] in which few media blocks may be discarded or delayed in a continuous playback process without significantly affecting the perceived quality.

1.1. Main contribution of the paper

The proposed negotiation mechanism reduces the QoS of several low-priority clients, by cutting out a small fraction of the assigned server capacity, to accept a new high-priority client and to achieve a higher net earning value. According to the simulation results, it can be seen that the negotiation mechanism is able to achieve a higher value of system performance.

In this paper, we propose performance analysis models for the mechanism. The analytical models can be used to dynamically determine the optimal setting of threshold values in real time upon the change of system load, in which the system workload is characterized by: (1) the arrival rate; (2) service rate; (3) reward rate; (4) penalty rate of each client. In this paper, the proposed analytical models are compared to the Stochastic Petri-Net (SPN) model, which takes exponential time (in terms of the number of states in the SPN model) to find optimal solutions. From the experimental results, it can be seen that the proposed analytical models find sub-optimal solutions that are very close to the optimal ones.

The rest of the paper is organized as follows. In Section 2, the description of the dynamic threshold-based algorithm with negotiation is given. The definition of the objective function is also defined there. In Section 3, the proposed analytical model is developed. Section 4 describes the numerical experiment results that are obtained from the analytical model and the Stochastic Petri-Net Package (SPNP) model. Finally, conclusions are drawn in Section 5.

2. System model

The server prioritizes client requests into different priority classes according to their importance to the system. The server adopts the reservation scheme in which a fraction of server capacity is allocated to a client throughout the existence of the request. Throughout the paper, we use the terms of “client” and “request” inter-exchangeably. The server capacity is divided into several partitions. One partition for each class of requests and possibly one common pool shared by multiple classes. Upon the arrival of a new client, the server checks the remaining capacity for the specific priority class of clients. If the remaining capacity is enough to serve a new request, it will be accepted; otherwise, a negotiation process may take place to determine if it can be accepted.

For the sake of explanation, we consider a system with two priority classes of requests here. Each class of requests is characterized by its arrival/departure rates and its reward/penalty values. Requests which provide high reward and penalty values [14,15] are considered as high-priority ones. Let the inter-arrival times of the high-priority and low-priority clients be exponentially distributed with the average times of $1/\lambda_{\mathrm{h}}$ and $1/\lambda_{\mathrm{l}}$, respectively. The inter-departure times of the high-priority and low-priority clients are exponentially distributed with the same service time of $1/\mu$. The proposed method is capable of handling different service times. However, we use the same service time for simplicity. Let the reward rate of high-priority and low-priority clients be v_{h} and v_{l}, respectively, with $v_{\mathrm{h}} \geq v_{\mathrm{l}}$, and the penalties be q_{h} and q_{l}, respectively, with $q_{\mathrm{h}} \geq q_{\mathrm{l}}$.

A server contains N capacity slots divided into three partitions: n_{h}, n_{l} and n_{m}, where $n_{\mathrm{m}} = N - n_{\mathrm{h}} - n_{\mathrm{l}}$. Capacity of n_{h} slots (referred as the high partition hereafter) is reserved for high-priority clients; n_{l} slots (referred as the low partition hereafter) for low-priority clients; while n_{m}, slots (referred as the common pool partition hereafter) are shared by all priority classes. We assume that all classes of clients have the same QoS requirements. Each capacity slot serves one client request initially, while the servicing of the low-priority clients could be degraded afterwards, if necessary. When a new client enters the system, the server checks the remaining capacity for the specific priority class. It is accepted if one such slot exists. Otherwise, the server checks the common pool. In other word, a new client can be assigned to the common pool only if the corresponding partition of the server capacity has no vacancy. A negotiation process starts if all slots in the common pool are occupied and the new coming request is a high-priority one.

The negotiation process reduces the QoS level of the low-priority requests in the common pool so as to make room for new arrival of high-priority requests. Each time α low-priority clients are chosen for degradation. Each such client is degraded by $1/\alpha$ and contributes $1/\alpha$ capacity slot. As a result, they make one slot in total. The total reward value of these degraded clients is $(\alpha-1)v_\mathrm{l}$, which is v_l less than the original total reward value contributed by them (i.e. αv_l) before degradation. A low-priority client is only degraded once. The degraded clients can be resumed to the normal QoS level upon the departure of a high-priority client. Note that no performance gain can be obtained if the negotiation process makes room for a new low-priority request. As stated above, the system gains extra value of $v_\mathrm{h}-v_\mathrm{l}$ from the negotiation process, for each newly admitted high-priority request.

Our objective function is the same as our performance index—the total *pay-off* rate, which is defined as the average amount of net earning received by the server per time unit. Let the system on average serve, per time unit, N_h: high-priority clients, N_l: low-priority ones, D_l: degraded low-priority ones, and reject, M_h: high-priority ones, and M_l: low-priority ones per time unit. The total pay-off rate can be obtained by the reward rate minus the penalty rate:

$$N_\mathrm{h}v_\mathrm{h}+N_\mathrm{l}v_\mathrm{l}+D_\mathrm{l}v_\mathrm{l}\frac{\alpha-1}{\alpha-M_\mathrm{h}q_\mathrm{h}-M_\mathrm{l}q_\mathrm{l}}. \tag{1}$$

The considered problem is formalized as finding an optimal set of threshold values under which the above objective function is maximized. Table 1 summarizes the notations used in the paper.

Table 1
Notations

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 successfully
v_l	Reward of a low-priority client if the client is serviced successfully
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
N	Total number of server capacity slots for servicing clients
n_h	Number of slots reserved for high-priority clients only, $0\le n_\mathrm{h}\le N$
n_l	Number of slots reserved for low-priority clients only, $0\le n_\mathrm{l}\le N$ and also $n_\mathrm{h}+n_\mathrm{l}\ge 0$
n_m	Number of slots that can be used to service either types of clients, $n_\mathrm{m}=N-n_\mathrm{h}-n_\mathrm{l}$
N_h	Number of high-priority clients served in the system per time unit, namely, the throughput of the high-priority clients
N_l	Number of low-priority clients served in the system per time unit, namely, the throughput of the low-priority clients
M_h	Number of high-priority clients rejected by the system per time unit
M_l	Number of low-priority clients rejected by the system per time unit
D_l	Number of degraded low-priority clients per time unit
α	Number of low-priority clients to be degraded to accommodate a new high-priority client

3. Finding the threshold setting

3.1. Two analytical methods

Consider a system with the threshold values (n_h, n_l, n_m). The arrival–departure process of high-priority clients served by the high partition of the n_h slots can be modeled as a $M/M/n_h/n_h$ queuing system. Similarly, the process of low-priority clients using the low partition of the n_l slots can be modeled as a $M/M/n_l/n_l$ queuing system. Therefore, the reward rates of the high- and low-priority clients served by the high and low partition are

$$\sum_{i=1}^{n_h} i\mu v_h \frac{(1/i!)(\lambda_h/\mu)^i}{\sum_{j=0}^{n_h}(1/j!)(\lambda_h/\mu)^j}, \tag{2}$$

and

$$\sum_{i=1}^{n_l} i\mu v_l \frac{(1/i!)(\lambda_l/\mu)^i}{\sum_{j=0}^{n_l}(1/j!)(\lambda_l/\mu)^j}. \tag{3}$$

Clients enter the common pool partition, only when there is no vacant slot in the corresponding partition. Therefore, the arrival rate of high- (low-) priority clients entering the common pool partition can be approximated as φ_h (φ_l). Namely

$$\varphi_h = \lambda_h \times \text{probability of having}\, n_h \text{ clients} = \lambda_h \frac{(1/n_h!)(\lambda_h/\mu)^{n_h}}{\sum_{j=0}^{n_h}(1/j!)(\lambda_h/\mu)^j}, \tag{4}$$

$$\varphi_l = \lambda_l \times \text{probability of having}\, n_l \text{ clients} = \lambda_l \frac{(1/n_l!)(\lambda_l/\mu)^{n_l}}{\sum_{j=0}^{n_l}(1/j!)(\lambda_l/\mu)^j}. \tag{5}$$

Let the probability that there are i high-priority clients and j low-priority clients served by the common pool partition be $P(i, j)$. The probability distribution of $P(i, j)$ can be approximated by the technique of reduced Markov chain [9]. In Eq. (6), the first term at the right-hand side indicates the probability of having i high-priority clients, and the second term indicates the probability of having j low-priority client, given i high-priority clients in the common pool partition

$$P(i, j) = \frac{(1/i!)(\varphi_h/\mu)^i}{\sum_{k=0}^{n_m}(1/k!)(\varphi_h/\mu)^k} \frac{(1/j!)(\varphi_l/\mu)^j}{\sum_{k=0}^{n_m-i}(1/k!)(\varphi_l/\mu)^k}. \tag{6}$$

Hence, the reward rate of the common pool partition is approximated as

$$\sum_{i=0}^{n_m}\sum_{j=0}^{n_m-i} P(i, j)(i\mu v_h + j\mu v_l). \tag{7}$$

Consider a state (i, j) in which $i + j = n_m$. Upon arrival of new high-priority clients, the negotiation process takes place to degrade the j low-priority clients to make room for the new high-priority arrivals. It can be seen that at most $\Omega(i)(= \lfloor (n_m - i)/\alpha \rfloor)$ slots can be squeezed out for the new high-priority clients. Since $n_m \geq i \geq 0$, it can be seen that $\lfloor n_m/\alpha \rfloor \geq \Omega(i) \geq 0$. Note that $\Omega(i)$ is defined as the number of slots that can be squeezed out for the arrival of new high-priority customers when i high-priority and

$n_m - i$ low-priority customers reside in the common pool. While the probability distribution of state $(i, n_m - i)$, namely $P(i, n_m - i)$ in Eq. (6), is sensitive to the relative arrival and service rates of high- and low-priority customers, $\Omega(i)$ is not a function of arrival rates. Two methods are developed in the following to approximate the pay-off rate obtained by negotiation.

3.1.1. Method 1

The arrival–departure process of high-priority clients, under a negotiation process, can be modeled as a $M/M/\Omega(i)/\Omega(i)$ queuing system. The arrival rate is $\Lambda(i) = \varphi_h P(i, n_m - i)$, where i is the number of high-priority clients in the common pool partition before negotiation is performed. We can see that $\Lambda(i)$ strongly depends on the value of $P(i, n_m - i)$ as shown in Eq. (6). Since $P(i, n_m - i)$ depends on the arrival rates and service rates of high- and low-priority clients, so does $\Lambda(i)$. Each time a new high-priority client is admitted, α low-priority clients are degraded and the penalty for the degradation is v_l. Note that the model is no longer valid after a high-priority client is admitted into the shared pool. It is because that the new arrival rate $\Lambda(i) = \varphi_h P(\,)$ is changed from $\varphi_h P(i, n_m - i)$ to $\varphi_h P(i + 1, n_m - i)$. However, to simplify the approximation computation, we assume the queuing model of $M/M/\Omega(i)/\Omega(i)$ is still valid during the whole negotiation process. The penalty rates of high-priority and low-priority clients are

$$\sum_{i=0}^{n_m} \Lambda(i) q_h \frac{(1/\Omega(i)^!)(\Lambda(i)/\mu)^{\Omega(i)}}{\sum_{j=0}^{\Omega(i)} (1/j!)(\Lambda(i)/\mu)^j}, \tag{8}$$

and

$$\sum_{i=0}^{n_m} P(i, n_m - i)\varphi_l q_l. \tag{9}$$

The reward rate of negotiation is

$$\sum_{i=0}^{n_m} \left[\sum_{\kappa=0}^{\Omega(i)} k\mu(v_h - v_l) \frac{(1/k!)(\Lambda(i)/\mu)^k}{\sum_{j=0}^{\Omega(i)} (1/j!)(\Lambda(i)/\mu)^j} \right]. \tag{10}$$

The overall pay-off rate can be approximated as (2) + (3) + (7) + (10) − (8) − (9).

3.1.2. Method 2

Another approach to modeling the negotiation process is to calculate the pay-off rate for each $P(i, n_m - i)$. The negotiation process is modeled as a $M/M/\Omega(i)/\Omega(i)$ queuing system with the arrival rate of φ_h. The penalty rate of high-priority clients can be expressed by Eq. (8), where $\Lambda(i)$ is replaced with φ_h. Similarly, the reward rate of high-priority clients can be expressed by Eq. (10), where $\Lambda(i)$ is replaced with φ_h. The pay-off rate of high-priority clients in the system with negotiation is

$$\sum_{i=0}^{n_m} P(i, n_m - i) \left[\sum_{k=0}^{\Omega(i)} k\mu(v_h - v_l) \frac{(1/k!)(\varphi_h/\mu)^k}{\sum_{j=0}^{\Omega(i)} (1/j!)(\varphi_h/\mu)^j} - \varphi_h q_h \frac{(1/\Omega(i)!)(\varphi_h/\mu)^{\Omega(i)}}{\sum_{j=0}^{\Omega(i)} (1/j!)(\varphi_h/\mu)^j} \right]. \tag{11}$$

Combining Eqs. (2), (3), (7), (9) and (11), the overall pay-off rate of a system using the dynamic threshold admission control with negotiation can be approximated as (2) + (3) + (7) + (11) − (9).

Table 2
Algorithm for finding the sub-optimal threshold setting

```
Partition(W) {
  Max_pay = −1;
  For $n_h$ = 0 to N do
    For $n_l$ = 0 to $N - n_h$ do
      $n_m = N - n_h - n_l$;
      pay = M1(W, $n_h$, $n_l$, $n_m$);
      if pay > Max_pay then
        Max_pay = pay;
        Assign ($n_h$, $n_l$, $n_m$) as optimal threshold setting;
    End for
  End for
  Return the optimal threshold setting;
}
```

3.2. *Applying the analytical methods*

The system pay-off rate with the threshold values (n_h, n_l, n_m) can be approximated by the two proposed methods, Method 1 and Method 2, in Section 3.1. Given a system workload, the two methods can be applied to find the sub-optimal threshold, as shown in Table 2. Let W be the system workload characterized by (N, λ_h, λ_l, μ, v_h, v_l, q_h, q_l, α). We can see that sub-routine *Partition*(W) calls Method 1/Method 2 (M1/M2 for short) iteratively until the maximum value is found. The input parameters to Methods 1 and 2 are the system workload W and the threshold values (n_h, n_l, n_m). Since the time complexity of *Partition*(W) is in polynomial time, finding the sub-optimal threshold setting can be performed in real time, upon the dynamic change of system workload.

4. Simulations

4.1. *SPN model*

Another approach to computing the value of the pay-off rate for a system is to use the SPNP [8], given a set of input parameters. The SPNP is a modeling tool developed in the Duke University for solving the SPN models. The SPN model of a system can be described in the C-based SPN Language (CSPL) of the SPNP. The steady-state solution of the SPN model can be solved by writing the SPNP output functions. Interested readers are suggested to study the SPNP manual [8] for further details.

The SPN model of the dynamic threshold scheme with negotiation (NEG) is illustrated in Figs. 1 and 2. The places RH, RL, and RS indicate the available capacity slots in the three partitions, the high, low, and common pool partitions, and have initially n_h, n_l, and αn_m tokens, respectively. In this model, one token represents one capacity slot and there are N tokens in the system. H and L represent the number of the high- and low-priority clients served by the high and low partition, respectively. SH and SL denote the number of the high- and low-priority clients served by the common pool partition, respectively. H, L, SH, and SL is set to zero initially. The place, SLL, indicates the number of degraded low-priority clients and is initialized to 0. One token in the high and low partitions represents one capacity slot in the

Places:

(In the high partition)

RH:	*mark*(RH) indicates the number of available tokens for high-priority clients.
H:	*mark*(H) indicates the number of high-priority clients being served ($mark$(RH)+ $mark$(H) = n_h)

(In the low partition)

RL:	*mark*(RL) indicates the number of available tokens for low-priority clients.
L:	*mark*(L) indicates the number of low-priority clients being served ($mark$(RL)+ $mark$(L) = n_l)

(In the common pool partition:)

RS:	*mark*(RS) indicates the number of tokens available for high- and low-priority clients.
SH:	*mark*(SH) indicates the number of tokens held by high-priority clients.
SL:	*mark*(SL) indicates the number of tokens held by low-priority clients.
SLL:	*mark*(SLL) indicates the number of tokens held by the degraded low-priority clients. ($mark$(RS) + $mark$(SH) + $mark$(SL) + $mark$(SLL) = $\alpha * n_m$)

Transition:	Rate Function:	Enabling function:
T1:	λ_h	$mark$(RH) = 0
T2:	λ_l	$mark$(RL) = 0
T3:	$mark$(SH) $* \mu / \alpha$	$mark$(SH) $\geq \alpha$
T4:	$mark$(SL) $* \mu / \alpha$	$mark$(SL) $\geq \alpha$
T5:	$mark$(SLL) $* \mu/(\alpha$-1)	$mark$(SLL) $\geq \alpha$-1
T6:	λ_h	$mark$(RS) = 0 & $mark$(RH) = 0 & $mark$(RL) = 0
T7:	(immediate transition)	$mark$(RS) > 0 & $mark$(SLL) $\geq \alpha$-1
T8:	λ_h	$mark$(RH) > 0
T9:	$mark$(H) $* \mu$	$mark$(H) > 0
T10:	λ_l	$mark$(RL) > 0
T11:	$mark$(L) $* \mu$	$mark$(L) > 0

Arc:	Multiplicity function:
RS → T1	α
T1 → SH	α
RS → T2	α
T2 → SL	α
SH → T3	α
T3 → RS	α
SL → T4	α
T4 → RS	α
SLL → T5	$\alpha - 1$
T5 → RS	$\alpha - 1$
SL → T6	$\alpha * \alpha$
T6 → SH	α
T6 → SLL	$\alpha * (\alpha - 1)$
SLL → T7	$\alpha - 1$
T7 → SL	α

Fig. 1. The interpretation of places, transitions, and arcs of the SPN model for the NEG.

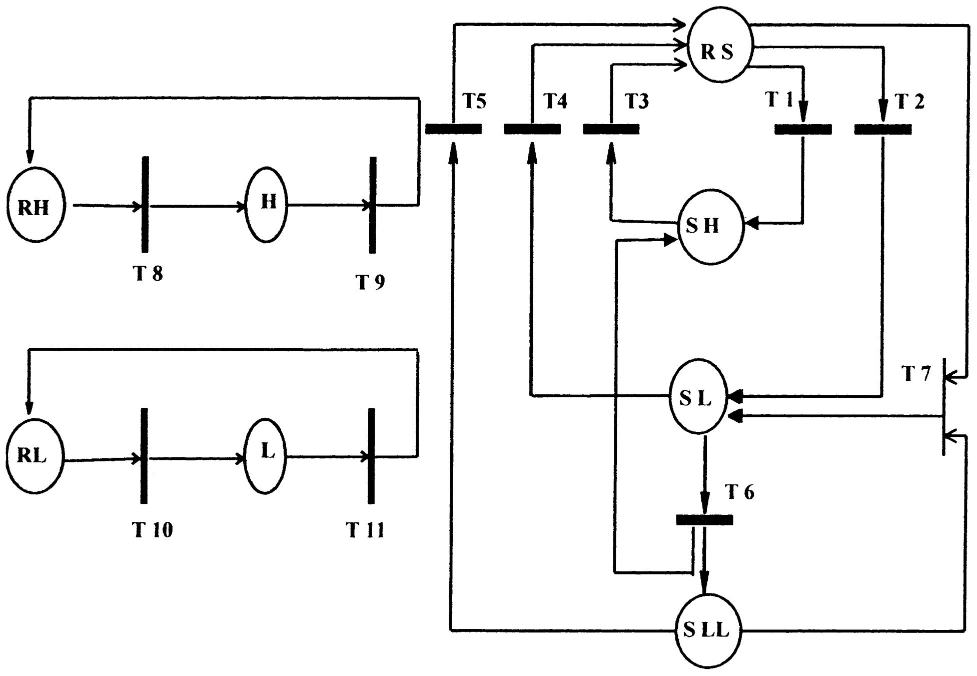


Fig. 2. The SPN model of the NEG.

system, while α tokens in the common pool partition represents one slot. Therefore, a client served by the common pool partition consumes α tokens by the transition T1 or T2, and returns α tokens to RS by T3 or T4 when leaving. When a negotiation process occurs, α low-priority clients (totally α' tokens) from SL are degraded. Each loses a token and they contribute α tokens in total. A new high-priority client is then able to be admitted and enters the place SH by the transition T6. The degraded low-priority clients (with total $\alpha(\alpha - 1)$ tokens) enter the place SLL by T6. A degraded client may leave the system and returns its tokens by T5. Degraded clients are resumed to the normal service level by T7, if RS contains available resource (i.e., tokens released by other clients). The interpretation of the places, arcs, and transitions is given in Fig. 1. Note that the three SPN sub-models for the high/low/common partitions are synchronized via the enabling functions and the markings of the places, as shown in Fig. 1.

The interpretation of key transitions in Fig. 1 is given as follows. Transition T1 with rate λ_h is enabled when the number of marking in the place RH is zero. It indicates that a high-priority client enters the common pool if the high partition is fully occupied. Therefore, the net arrival rate of high-priority clients to the common pool can be approximated by $\lambda_h \times$ probability (high partition is full), which is equivalent to the terms in Eq. (4). Similar relation can be observed among transition T2, λ_l, and Eq. (5). Interested readers may refer to Fig. 1 and Section 3 for further details.

The system pay-off rate with the threshold values (n_h, n_l, n_m) can be exactly computed by using the SPN model. Therefore, given a system workload, the SPN model can be applied to find the optimal threshold. The algorithm for using the SPN model to find the optimal setting is similar to the one shown in Table 2, and it is omitted herein. However, the time complexity of using the SPN model is in exponential time. It is infeasible to apply the SPN model to the dynamic workload change environment.

4.2. Numerical results

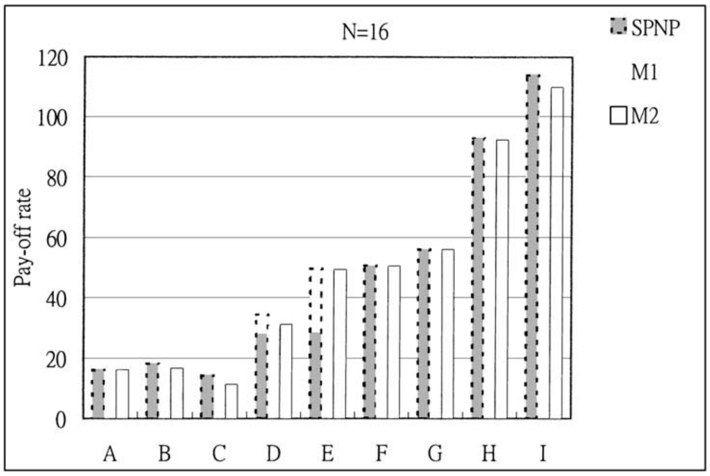
Consider an example of building an on-demand multimedia system on CATV network [17]. The system delivers the multimedia services via the hybrid fiber coaxial (HFC) access network. Clients with different priority levels enter the system via QoS manager. The main responsibility of the QoS manager is admission control and dynamic resource (i.e. network bandwidth and/or server storage) allocation. An array of 64 disks each with a storage capacity of 1 GB can be implemented in the server, as indicated from the experiment results [6].

Continuous media blocks with 512 KB each of a video stream are randomly stored on the disk. The playback rate is assumed to be 30 frames/s per client request. For such physical configuration, the maximum number of clients that concurrently exist in the system was found to be near 16 if strict deterministic admission control is performed. The number of clients could be up to 32 if video compression is applied. According to our cost model, high-priority clients contribute higher reward and incur higher penalty if rejected. Workload characteristics of such a system are changeable such that a static admission control algorithm is infeasible and unable to adapt to the run time changes.

Admission control with NEG can be implemented in a multimedia system as stated above. One challenge facing the NEG algorithm is dynamic partitioning of the system resource as workload changes. An optimal setting of the threshold values for any workload conditions shall be found so as to maximize the system pay-off rate. One way to dynamic partitioning is to identify the possible workload conditions before the system is up for service. Time complexity is the main concern of solving the SPN model by the SPNP. The experiments are run on a SUN Ultra-1 model 140 machine equipped with a 143 MHz UltraSPARC processor, 32 MB memory, and 2.1 GB FAST SCSI-2 hard disk. On average, it takes 94 and 6678 s (i.e. approximately 1 h and 50 min) to find out the optimal settings, for $N = 16$ and 32, respectively. For such a reason, the optimal threshold values are obtained from the SPNP tool before run time, for each identified workload. The optimal settings are maintained in a table such that the QoS manager is capable of looking up the table to accordingly re-configure the resource partition at run time, upon a workload change. The limitation of such an approach is the contents of the look-up table. In an event of a sudden change that was not identified before hand, the SPNP-approach is unable to respond in real time. Consequently, the SPNP-approach falls apart.

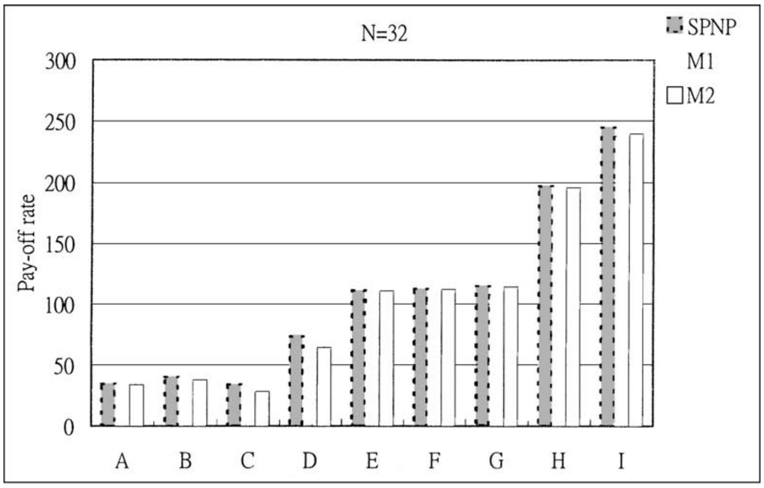
On the other hand, the analytical approaches are capable of finding sub-optimal solutions in real time, as workload changes. The optimal threshold values found by Methods 1 and 2 could be different from those by the SPNP. Let the optimal settings found by the SPNP, Method 1, and Method 2 be x_1, x_2, and x_3 respectively, given a workload condition. Note that x_2 (or x_3) being the optimal setting of Method 1 (or 2) means that the pay-off value of x_2 (or x_3) calculated by the method is the maximum. However, it is not true in the real case. The true pay-off rate of x_2 should be the one obtained by solving the SPN model when the partition is specified according to the values in x_2. Therefore, the maximum pay-off values by Methods 1 and 2 are calculated by mapping their optimal threshold values to the SPNP.

Experiment results are illustrated in Figs. 3 and 4. The combined threshold settings along with the pay-off rates are also shown in Tables 3 and 4. They demonstrate that the system performance (pay-off rate) by the analytical methods is very close to that by the SPNP. Method 2 (short for M2 in the figures) performs slightly better than Method 1 (short for M1 in the figures). For $N = 16$, the average performance difference between the SPNP and M1 is 3.88%, while that between the SPNP and M2 is 2.83%. For $N = 32$, the difference between the SPNP and M1 is 4.53% on average, while that between the SPNP



System Parameters of $(\lambda_h, \lambda_l, \mu, v_h, v_l, q_h, q_l, \alpha)$		
A = (5, 10, 1, 2, 1, 2, 1, 2)	D = (5, 10, 1, 5, 1, 2, 1, 2)	G = (5, 10, 1, 10, 1, 2, 1, 2)
B = (10, 10 ,1, 2, 1, 2, 1, 2)	E = (10, 10 ,1, 5, 1, 2, 1, 2)	H = (10, 10 ,1, 10, 1, 2, 1,2)
C = (15, 10, 1, 2, 1, 2, 1, 2)	F = (15, 10, 1, 5, 1, 2, 1, 2)	I = (15, 10, 1, 10, 1, 2, 1, 2)

Fig. 3. Experimental results for $N = 16$.



System Parameters $(\lambda_h, \lambda_l, \mu, v_h, v_l, q_h, q_l, \alpha)$		
A = (10, 20, 1, 2, 1, 2, 1, 2)	D = (10, 20, 1, 5, 1, 2, 1, 2)	G = (10, 20, 1, 10, 1, 2, 1, 2)
B = (20, 20 ,1, 2, 1, 2, 1, 2)	E = (20, 20 ,1, 5, 1, 2, 1, 2)	H = (20, 20 ,1, 10, 1, 2, 1,2)
C = (30, 20, 1, 2, 1, 2, 1, 2)	F = (30, 20, 1, 5, 1, 2, 1, 2)	I = (30, 20, 1, 10, 1, 2, 1, 2)

Fig. 4. Experimental results for $N = 32$.

Table 3
The actual threshold and the pay-off rate for $N = 16$ cases

$N = 16$	Pay-off rate			Threshold setting (n_h, n_l, n_m)		
	SPNP	Method 1	Method 2	SPNP	Method 1	Method 2
$A = 5, 10, 1, 2, 1, 2, 1$	16.06934	15.84194	15.95887	0, 0, 16	0, 4, 12	1, 0, 15
$B = 10, 10, 1, 2, 1, 2, 1$	18.13524	18.13524	16.66035	0, 0, 16	0, 0, 16	8, 0, 8
$C = 15, 10, 1, 2, 1, 2, 1$	14.34474	13.70442	11.32386	0, 0, 16	8, 0, 8	16, 0, 0
$D = 5, 10, 1, 5, 1, 2, 1$	34.31614	30.64462	31.04892	8, 0, 8	0, 4, 12	0, 0, 16
$E = 10, 10, 1, 5, 1, 2, 1$	49.63583	46.17084	49.45205	8, 0, 8	0, 0, 16	7, 0, 9
$F = 15, 10, 1, 5, 1, 2, 1$	50.67544	49.60181	50.48349	10, 0, 6	0, 0, 16	12, 0, 4
$G = 5, 10, 1, 10, 1, 2, 1$	56.01489	53.39420	56.01489	0, 0, 16	0, 4, 12	0, 0, 16
$H = 10, 10, 1, 10, 1, 2, 1$	92.89684	92.89684	92.50624	0, 0, 16	0, 0, 16	6, 0, 10
$I = 15, 10, 1, 10, 1, 2, 1$	113.97154	108.36357	110.00581	16, 0, 0	0, 0, 16	11, 0, 5

Table 4
The actual threshold and the pay-off rate for $N = 32$ cases

$N = 32$	Pay-off rate			Threshold setting (n_h, n_l, n_m)		
	SPNP	Method 1	Method 2	SPNP	Method 1	Method 2
$A = 10, 20, 1, 2, 1, 2, 1$	34.66556	34.34261	34.32542	0, 0, 32	0, 12, 20	4, 5, 23
$B = 20, 20, 1, 2, 1, 2, 1$	40.25709	40.25709	37.91433	0, 0, 32	0, 0, 32	16, 0, 16
$C = 30, 20, 1, 2, 1, 2, 1$	34.17334	33.06303	28.44801	0, 0, 32	18, 0, 14	32, 0, 0
$D = 10, 20, 1, 5, 1, 2, 1$	73.46047	63.98188	64.66544	15, 4, 13	0, 13, 19	0, 0, 32
$E = 20, 20, 1, 5, 1, 2, 1$	111.13932	98.54519	110.64375	17, 0, 15	0, 1, 31	15, 0, 17
$F = 30, 20, 1, 5, 1, 2, 1$	112.34387	110.45593	112.13882	24, 0, 8	0, 0, 32	26, 0, 6
$G = 10, 20, 1, 10, 1, 2, 1$	114.66524	107.16476	114.66524	0, 0, 32	0, 14, 18	0, 0, 32
$H = 20, 20, 1, 10, 1, 2, 1$	197.30296	194.27755	196.17262	0, 0, 32	0, 1, 31	13, 0, 19
$I = 30, 20, 1, 10, 1, 2, 1$	245.34383	237.59357	240.41279	32, 0, 0	0, 0, 32	24, 0, 8

and M2 is 2.48%. The performance differences between the SPNP and the approximation methods are within a reasonable range.

5. Conclusions

In this paper, we have investigated the admission control problem for the systems with two classes of client requests and the analytical model. In the cost model, each class of request has its reward and penalty to the system. High-priority requests are associated with high reward and penalty values. We have proposed an admission control algorithm with negotiation mechanism and evaluate its performance. Negotiation attempts to accept high-priority requests under heavy and over loaded systems, lowering the service requirements of some low-priority requests. The SPN model is used to find optimal solutions and the analytical approaches are developed to find sub-optimal ones. The results show that the sub-optimal solutions found by the proposed analytical methods are very close to optimal ones. Therefore, a multimedia server can exploit the proposed performance-evaluation methods to dynamically adjust threshold values based on the characteristics of the workload in order to achieve high system performance.

Some future research areas include: (a) extending negotiation to a system with multiple priority classes; (b) changing the mandatory negotiation mechanism to a voluntary degradation one, in which the low-priority clients have options either to keep their QoS levels or to accept the degradation in an altruism fashion.

References

[1] S. Ramanathan, P.V. Rangan, Architecture for personalized multimedia, IEEE Multimedia 1 (1) (1994) 37–46.
[2] H.M. Vin, A. Goyal, P. Goyal, Algorithms for designing multimedia servers, Comput. Commun. 18 (1995) 192–203.
[3] E. Chang, A. Zakhor, Cost analysis for VBR video servers, IEEE Multimedia 3 (3) (1996) 56–71.
[4] P. Rangan, H.M. Vin, Designing file systems for digital video and audio, in: Proceedings of the 12th ACM Symposium on Operating Systems, 1991.
[5] H.M. Vin, P. Goyal, A. Goyal, A statistical admission control algorithm for multimedia servers, in: Proceedings of the ACM International Conference on Multimedia, San Francisco, 1994.
[6] I. Chen, C. Chen, Threshold-based admission control policies for multimedia servers, Comput. J. 39 (9) (1996) 1–10.
[7] H.M. Vin, A. Goyal, P. Goyal, An observation-based admission control algorithms for multimedia servers, in: Proceedings of the First IEEE International Conference on Multimedia Computing and Systems, Boston, 1995.
[8] K.S. Trivedi, G. Ciardo, J.K. Muppala, Manual for the SPNP Package, Department of Electrical Engineering, Duke University, Durham, NC, 1991.
[9] L. Kleinrock, Queuing Systems Theory, vol. 1, Wiley, New York, 1975.
[10] M. Chen, H. Hsiao, C. Li, P. Yu, Using rotational mirrored declustering for replica placement in a disk-array-based video server, ACM Multimedia Syst. 5 (1997) 371–379.
[11] H.M. Vin, P. Rangan, Designing a multi-user HDTV storage server, IEEE J. Select. Areas Commun. 11 (1) (1993) 153–164.
[12] E. Oomoto, K. Tanaka, OVID: design and implementation of a video-object database system, IEEE Trans. Knowled. Data Eng. 5 (1993) 629–643.
[13] Y. Oyang, C. Wen, C. Cheng, C. Lee, J. Li, A multimedia storage system for on-demand playback, IEEE Trans. Consumer Elect. 41 (1995) 53–64.
[14] A. Bestavros, S. Braoudakis, Value-cognizant speculative concurrency control, in: Proceedings of the International Conference on Very Large Databases, September 1995.
[15] C. Locke, Best effort decision making for real-time scheduling, Ph.D. Thesis, Carnegie-Mellon University, Department of Computer Science, PA, 1986.
[16] C.W. Mercer, S. Savage, H. Tokuda, Processor capacity reserves: operating system support for multimedia applications, in: Proceedings of the First IEEE International Conference on Multimedia Computing and Systems, Boston, 1994, pp. 90–99.
[17] Y. Kuo, W. Lee, W. Cheng, M. Horng, The implementation of intelligent service navigator for virtual club multimedia service system on CATV network, in: Proceedings of the Workshop on Distributed System Technologies and Applications, Taiwan, 1998, pp. 393–401.