1998 International Computer Symposium
Workshop on Computer Networks, Internet, and Multimedia
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

Dynamic-threshold Admission Control with Negotiation in Multimedia Servers

Sheng-Tzong Cheng and Chi-Ming Chen
Department of Computer Science and Information Engineering,
National Cheng Kung University, Tainan, Taiwan, R.O.C.
stcheng@mail.ncku.edu.tw
Ing-Ray Chen
Department of Computer Science
Institute of Virginia Technology,
Virginia, USA.

Abstract

A novel threshold-based admission control algorithm with negotiation for two priority classes of requests is proposed in this 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. Given the characteristics of the system workload, the proposed algorithm finds the best partitions, optimizing the system performance based on the objective function of the total reward minus the total penalty. The negotiation mechanism will reduce 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 approximation approaches are developed to find sub-optimal settings. The experiment results show that the proposed algorithm performs better than one without negotiation mechanism, and that the sub-optimal solutions found by the proposed approximation approaches are very close to optimal ones. The approximation approaches enable the algorithm to dynamically adjust the threshold values, based on the characteristics of the system workload, to achieve higher system performance.

1 INTRODUCTION

Delivering multimedia streams with QoS requirements to viewers is one crucial issue in designing a multimedia system. 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 later 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 above research does not consider different priorities of client requests. 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. 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.

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. In this paper, we propose the dynamic threshold-based algorithm with *negotiation* mechanism that will reduce 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.

2 SYSTEM MODEL

The server prioritizes client requests into different priority classes according to their importance to the system. 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.

We consider a system with two priority classes of requests. Each class of requests is characterized by its arrival/departure rates and its reward/penalty values. Requests provide high reward and penalty [14,15] are considered as high-priority ones. Let the inter-arrival times of the high-priority and low-priority clients be

This work is sponsored by the National Science Council, Taiwan, R.O.C. under the contract NSC 87-2213-E006-075

exponentially distributed with the average times of $1/\lambda_h$ and $1/\lambda_l$, 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 $v_h \geq v_l$, and the penalties be q_h and q_l, respectively, with $q_h \geq q_l$.

A server contains n capacity slots divided into three partitions: n_h, n_l and n_m, where $n_m = N - n_h - n_l$. 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 and hence each capacity slot serves one client request. When a new client enters the system, the server checks the remaining capacity for the specific priority class. 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 $(\alpha-1)* v_l$, which is v_l less than the original total reward value contributed by them (i.e. $\alpha* v_l$) before degradation. For the sake of service quality, 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 $v_h - v_l$ 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, and 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 as shown in (1). The proposed 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.

$$N_h v_h + N_l v_l + D_l v_l*(\alpha-1)/\alpha - M_h q_h - M_l q_l \qquad (1)$$

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, $0 \leq n_h \leq N$
n_l	Number of slots reserved for low-priority clients only, $0 \leq n_l \leq N$ and also $n_h + n_l \geq 0$
n_m	Number of slots that can be used to service either types of clients, $n_m = N - n_h - n_l$
N_h	Number of high-priority clients served in the system per time unit
N_l	Number of low-priority clients served in the system per time unit
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

Table 1. Notation

3 STOCHASTIC PETRI NET MODELS

The value of the pay-off rate for a system can be obtained by the Stochastic Petri Net Package (SPNP) [8], given a set of input parameters. The SPNP is a modeling tool developed in the Duke University for solving the Stochastic Petri Net (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 without negotiation (NoNEG) is plotted in Figure 1. 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 n_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 SPN model of the dynamic-threshold scheme

with negotiation (NEG) is shown in Figure 2. The notations and their initial values are the same as those in Figure 1, except that RS is initialized to $\alpha * n_m$. The new 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 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^2 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 free resource (i.e., tokens released by other clients).

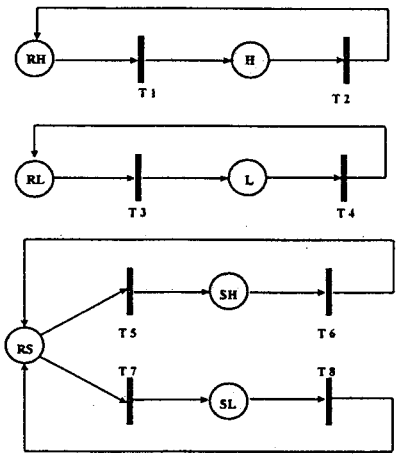


Figure 1. The SPN model of the NoNEG

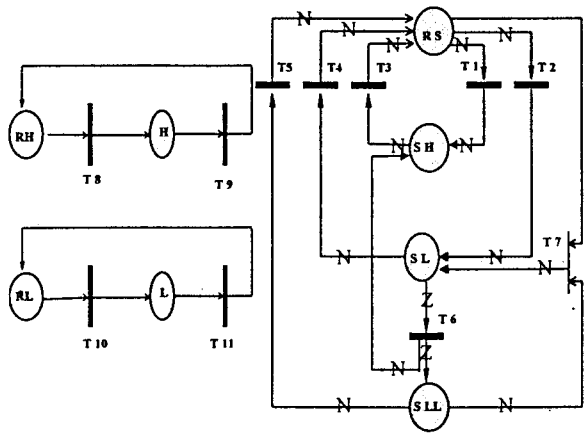


Figure 2. The SPN model of the NEG

3.1 Experimental Results

To evaluate the performance of the two algorithms, we define two comparison measurements: the best-case and average-case *gain ratio*. The best-case gain ratio is defined as

$$\max\left(\frac{\mathrm{NEG}(x)-\mathrm{NoNEG}(x)}{\mathrm{NoNEG}(x)}, \forall x\right),$$

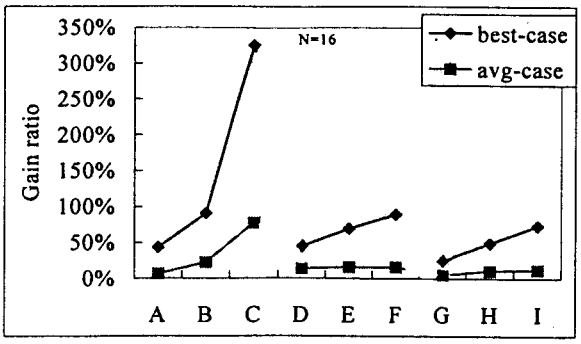
where x is one of the possible partitioning (i.e. the combinations of the threshold values)of the server capacity, and NEG(x) and NoNEG(x) indicate the pay-off rates, given the partition configuration x, obtained by the NEG and NoNEG algorithms, respectively. The average-case gain ratio is defined as

$$\frac{\sum_{x} \frac{\mathrm{NEG}(x)-\mathrm{NoNEG}(x)}{\mathrm{NoNEG}(x)}}{C(N+2,2)}.$$

The input parameters to the SPN models of the NEG and NoNEG algorithms are the arrival and departure rates, λ_h, λ_l and μ, the reward and penalty parameters v_h, v_l, q_h, and q_l. The pay-off rate with the threshold values (n_h, n_l, n_m) can be obtained by the following steps. (1) Model the system based on the SPN model; (2) calculate the values of N_h, N_l, M_h, M_l by the SPNP; and (3) compute the pay-off rate by equation 1.

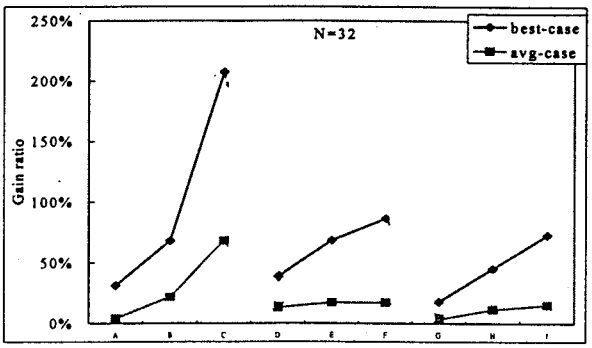
Figures 3 and 4 show the comparison results for N = 16 and 32 under various workload conditions. Two curves are plotted: one for the best-case gain ratio and the other for the average-case one. A system is nearly saturated when the utilization value is round 15/16, and is over-saturated when it is greater than 1. The average-case and best-case gain ratios rise as the workload increases, as shown in each set. It illustrates that (1) the negotiation mechanism allows a system to achieve a higher pay-off rate under heavy and over-loaded situations and (2) it is beneficial to apply negotiation in heavy and over-loaded systems, especially when the arrival rate of high-priority clients is large.

Optimal threshold value setting of (n_h, n_l, n_m) for the NoNEG and NEG algorithms can be exploited to maximize the pay-off rate for a system, given the workload characteristics. The NoNEG and NEG algorithms find out the optimal setting by enumerating all possible combinations of the threshold value settings, calculating the pay-off rate for each combination, and selecting the combination with the maximum pay-off rate. Tables 2 and 3 illustrate the optimal settings for the NoNEG and NEG algorithms under various workloads.



System Parameters $(\lambda_h, \lambda_l, \mu, v_h, v_l, q_h, q_l, \alpha)$		
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)

Figure 3 Performance comparison 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)

Figure 4. Performance comparison for N =32

As shown below, optimal threshold values can be found based on the proposed SPN model. However, analyzing a SPN model is very time-consuming. Therefore, we propose two approximation methods based on queuing analysis. The goal of the approximation methods is to enable the server adaptively configure the resource capacity according to the run-time workload.

System Parameters (for N = 16)	*No-Negotiation (by SPNP)*		*Negotiation (by SPNP)*	
$(\lambda_h, \lambda_l, \mu, v_h, v_l, q_h, q_l, \alpha)$	Optimal (n_h, n_m, n_l)	Pay-off rate	Optimal (n_h, n_m, n_l)	Pay-off rate
A=(5,10,1,2,1,2,1,2)	2,14,0	14.25	0,16,0	16.07
B=(10,10,1,2,1,2,1,2)	9,7,0	13.59	0,16,0	18.14
C=(15,10,1,2,1,2,1,2)	16,0,0	11.32	0,16,0	14.34
D=(5,10,1,5,1,2,1,2)	5,11,0	27.77	8,8,0	34.32
E=(10,10,1,5,1,2,1,2)	13,3,0	40.26	8,8,0	49.64
F=(15,10,1,5,1,2,1,2)	16,0,0	49.82	10,6,0	50.68
G=(5,10,1,10,1,2,1,2)	7,9,0	51.28	0,16,0	56.01
H=(10,10,1,10,1,2,1,2)	15,1,0	87.67	0,16,0	92.90
I=(15,10,1,10,1,2,1,2)	16,0,0	113.97	16,0,0	113.97

Table 2. Optimal pay-off rates and threshold values for N =16

System Parameters (for N = 16)	*No-Negotiation (by SPNP)*		*Negotiation (by SPNP)*	
$(\lambda_h, \lambda_l, \mu, v_h, v_l, q_h, q_l, \alpha)$	Optimal (n_h, n_m, n_l)	Pay-off rate	Optimal (n_h, n_m, n_l)	Pay-off rate
A=(10,20,1,2,1,2,1,2)	5,27,0	32.41	0,32,0	34.67
B=(20,20,1,2,1,2,1,2)	19,13,0	32.00	0,32,0	40.26
C=(30,20,1,2,1,2,1,2)	31,1,0	28.46	0,32,0	34.17
D=(10,20,1,5,1,2,1,2)	10,22,0	60.58	15,13,4	73.46
E=(20,20,1,5,1,2,1,2)	24,8,0	87.42	17,15,0	111.14
F=(30,20,1,5,1,2,1,2)	32,0,0	109.78	24,8,0	112.34
G=(10,20,1,10,1,2,1,2)	12,20,0	108.69	0,32,0	114.67
H=(20,20,1,10,1,2,1,2)	27,5,0	183.86	0,32,0	197.30
I=(30,20,1,10,1,2,1,2)	32,0,0	245.34	32,0,0,	245.34

Table 3. Optimal pay-off rates and threshold values for N =32

4. APPROXIMATION 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

$$\sum_{i=1}^{n_h} i\mu v_h \times \frac{\frac{1}{i!}\left(\frac{\lambda_h}{\mu}\right)^i}{\sum_{j=0}^{n_h}\frac{1}{j!}\left(\frac{\lambda_h}{\mu}\right)^j} \qquad (2)$$

$$\sum_{i=1}^{n_l} i\mu v_l \times \frac{\frac{1}{i!}\left(\frac{\lambda_l}{\mu}\right)^i}{\sum_{j=0}^{n_l}\frac{1}{j!}\left(\frac{\lambda_l}{\mu}\right)^j}. \qquad (3)$$

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,

$$\varphi_h = \lambda_h \times probability\ \text{of having}\ n_h\ \text{clients} = \lambda_h \times \frac{\frac{1}{n_h!}\left(\frac{\lambda_h}{\mu}\right)^{n_h}}{\sum_{j=0}^{n_h}\frac{1}{j!}\left(\frac{\lambda_h}{\mu}\right)^j}. \qquad (4)$$

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

Let the probability that there are i high-priority clients and j low-priority clients served by the common pool partition be $P(i, j)$ as shown in equation (6), the reward rate of the common pool partition is approximated as equation (7).

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

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

Consider a state (i, j) in which $i + j = n_m$. Upon arrival of a new high-priority client, the negotiation process takes place to degrade the j low-priority clients to make room for the new high-priority arrival. 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. Two methods are developed in the following to approximate the pay-off rate obtained by negotiation.

Method I

The arrival-departure process of high-priority clients, under a negotiation process, can be modeled as a M/M/Ω(i)/Ω(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. Each time a new high-priority client is admitted, α low-priority clients are degraded and the penalty for the degradation is v_l. The penalty rates of high-priority and low-priority clients are

$$\sum_{i=0}^{n_m}\Lambda(i)\times q_h\times\frac{\frac{1}{\Omega(i)!}\left(\frac{\Lambda(i)}{\mu}\right)^{\alpha(i)}}{\sum_{j=0}^{\alpha(i)}\frac{1}{j!}\left(\frac{\Lambda(i)}{\mu}\right)^j} \quad (8) \text{ and}$$

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

The reward rate of negotiation is

$$\sum_{i=0}^{nm}\left[\sum_{k=0}^{\Omega(i)}k\,\mu\times(v_h-v_l)\frac{\frac{1}{k!}\left(\frac{\Lambda(i)}{\mu}\right)^k}{\sum_{j=0}^{\Omega(i)}\frac{1}{j!}\left(\frac{\Lambda(i)}{\mu}\right)^j}\right] \quad (10)$$

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

Method II

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/Ω(i)/Ω(i) queuing system with the arrival rate of φ_h. The penalty rate of high-priority clients can be expressed by equation 8, where $\Lambda(i)$ is replaced with φ_h. Similarly, the reward rate of high-priority clients can be expressed by equation 10, where $\Lambda(i)$ is replaced with φ_h. The pay-off rate of high-priority clients in the system with negotiation is

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

Combining equations 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).

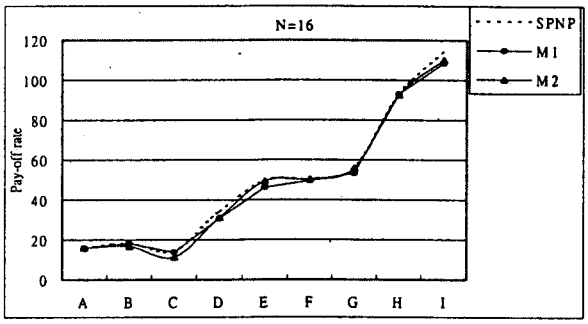
5. NUMERICAL EXPERIMENTS

Admission control with negotiation (NEG) can be implemented in a multimedia system. 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 143MHz UltraSPARC processor, 32MB memory, and 2.1GB FAST SCSI-2 hard disk. On average, it takes 94 and 6678 seconds (i.e. approximately one hour and 50 minutes) to find out the optimal settings, for N = 16 and N = 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 approximation approaches are capable of finding sub-optimal solutions in real time, as workload changes. The optimal threshold values found by Methods I and II could be different from those by the SPNP. Let the optimal settings found by the SPNP, Methods I and II 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 I (or II) 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

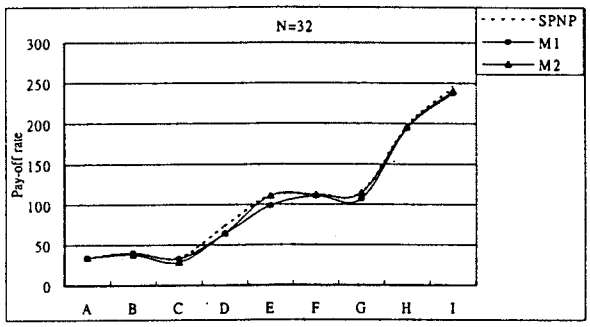
Methods I and II are calculated by mapping their optimal threshold values to the SPNP.

Experiment results are illustrated in Figures 5 and 6. They demonstrate that the system performance (pay-off rate) by the approximation methods is very close to that by the SPNP. Method II (short for M2 in the figures) performs slightly better than Method I (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 and M2 is 2.48%. The performance differences between the SPNP and the approximation methods are within a reasonable range.



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)

Figure 5. Approximation results for N =16.



System Parameters of $(\lambda_h, \lambda_l, \mu, V_h, V_l, q_h, q_l, \alpha)$		
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)

Figure 6. Approximation results for N =32.

6. CONCLUSION

In this paper, we have investigated the admission control problem for the systems with two classes of client requests and the cost 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 investigate 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 experimental results demonstrate that the negotiation mechanism can significantly improve the system performance. The Stochastic Petri-Net model is used to find optimal solutions and the approximation approaches are developed to find sub-optimal ones. The results show that the sub-optimal solutions found by the proposed approximation methods are very close to optimal ones. Therefore, a multimedia server can exploit the approximation 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, and (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] Ramanathan, S. and Rangan, P. V. (1994) Architecture for personalized multimedia. *IEEE Multimedia*, Spring, pp. 37-46.

[2] Vin, H. M., Goyal, A., and Goyal, P. (1995) Algorithms for designing multimedia servers. *Computer Communications*, 18, pp. 192-203.

[3] Chang, E. and Zakhor, A. (1996) Cost analysis for VBR video servers. *IEEE Multimedia*, Winter, pp. 56-71.

[4] Rangan, P. and Vin, H. M. (1991) Designing File Systems for Digital Video and Audio. *12th ACM Symposium on Operating Systems.*

[5] Vin, H. M., Goyal, P., and Goyal, A. (1994) A statistical admission control algorithm for multimedia servers. *ACM Intl. Conference on Multimedia*, San Francisco.

[6] Chen, I. and Chen, C. (1996) Threshold-based admission control policies for multimedia servers. *Computer Journal*, 39(9), pp. 1-10.

[7] Vin, H. M., Goyal, A. and Goyal, P. (1995) An observation-based admission control algorithms for multimedia servers. *1st IEEE Intl. Conference on multimedia computing and systems*, Boston.

[8] Trivedi, K. S., Ciardo, G. and Muppala, J. K. (1991) Manual for the SPNP Package. *Dept. of Electrical Engineering*, Duke University, Durham, NC.

[9] Kleinrock, L. (1975) *Queuing Systems, Vol. 1: Theory.* John Wiley and Sons.

[10] Chen, M., Hsiao, H., Li, C., and Yu, P. (1997) Using rotational mirrored declustering for replica placement in a disk-array-based video server, *ACM Multimedia Systems*, Vol. 5, December, pp. 371-379.

[11] Vin, H. M., Rangan P. (1993) Designing a multi-user

HDTV storage server. IEEE Journal on Selected Areas in Communications, 11(1), pp. 153-164.

[12] Oomoto, E. and Tanaka, K. (1993) OVID: Design and implementation of a video-object database system. *IEEE Transactions on Knowledge and Data Engineering*, 5, pp. 629 –643.

[13] Oyang, Y., Wen, C., Cheng, C., Lee, C. and Li, J. (1995) A multimedia storage system for on-demand playback. *IEEE Transactions on Consumer Electronics*, 41, pp. 53-64.

[14] Bestavros, A. and Braoudakis, S. (1995) Value-congnizant speculative concurrency control. *The Intl. Conference on Very Large Databases*. September.

[15] Locke, C. (1986) Best effort decision making for real-time scheduling. Ph.D. thesis, Carnegie-Mellon University, Dept. of Computer Science, PA.

[16] Mercer, C. W., Savage, S., and Tokuda, H. (1994) Processor capacity reserves: Operating system support for multimedia applications, *1st IEEE Intl. Conference on Multimedia Computing and Systems*, Boston, pp. 90-99.

[17] Kuo, Y., Lee, W., Cheng, W., and Horng, M. (1998) The Implementation of Intelligent Service Navigator for Virtual Club Multimedia Service System on CATV Network, *1998 Workshop on Distributed System Technologies & Applications,* Taiwan, pp.393-401.