ELSEVIER

Information Processing Letters 83 (2002) 337–344

Information Processing Letters

www.elsevier.com/locate/ipl

A self-adjusting quality of service control scheme

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

[a] *Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan*
[b] *Department of Computer Science, Virginia Tech., Northern Virginia Graduate Center, Falls Church, VA 22043, USA*

Received 8 August 2000; received in revised form 21 September 2001

Communicated by F.B. Schneider

Abstract

We propose and analyze a self-adjusting Quality of Service (QoS) control scheme with the goal of optimizing the system reward as a result of servicing different priority clients with varying workload, QoS and reward/penalty requirements. Our scheme is based on resource partitioning and designated "degrade QoS areas" such that system resources are partitioned into priority areas each of which is reserved specifically to serve only clients in a corresponding class with no QoS degradation, plus one "degraded QoS area" into which all clients can be admitted with QoS adjustment being applied only to the lowest priority clients. We show that the best partition is dictated by the workload and the reward/penalty characteristics of clients in difference priority classes. The analysis results can be used by a QoS manager to optimize the system total reward dynamically in response to changing workloads at run time. We demonstrate the validity of our scheme by means of simulation and comparing the proposed QoS self-adjusting scheme with those that do not use resource partitioning or designated degraded QoS areas. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Quality of service (QoS); QoS negotiation; Multimedia systems; Telecommunication systems; Admission control; Resource reservation; Reward optimization; Simulation; Performance analysis; Scheduling; Performance evaluation

1. Introduction

Quality of Service (QoS) control is an important issue in multimedia/telecommunication systems designed to provide continuous services to clients based on their QoS demands. To date, there are two approaches by which QoS control can be implemented. One approach is based on adaptive, distributed control wherein each client monitors the QoS received and automatically increases or decreases its resource requirement according to actual QoS level delivered to it and also by the amount of resources sensed available in the system. Another approach is based on a priori *resource reservation* wherein a centralized QoS control manager is used to interact with clients. Whenever a client requests a service of the system, it negotiates its QoS requirement with the QoS manager which checks its resources to make sure the client's QoS requirement can be satisfied before admitting the client into the system. This paper concerns the second approach.

Over the past few years, there has been substantial research effort in the area of QoS control [1,6,7]. A general approach is to reserve a portion of system resources to serve requests with degraded QoS

[*] Corresponding author.
E-mail addresses: stcheng@csie.ncku.edu.tw (S.-T. Cheng), irchen@cs.vt.edu (I.-R. Chen).

so as to tradeoff service quality for improved rejection rate. For example, Marsan et al. [7] demonstrate that reserving free channels to serve handovers and degraded multimedia calls can significantly reduce the handover failure probability in cellular networks at the expense of a higher new call rejection rate. Although these studies reveal useful tradeoffs, the issue of *how much* resources should be reserved for QoS control remains unsolved since optimizing one performance metric (e.g., low handover probability) may compromise another (e.g., new call rejection rate). Lee and Sabata [5] propose the concept of *benefit functions* and *resource demand functions* associated with application requests, characterizing each application with a range of QoS requirements and its corresponding "benefit" values with which the application brings to the system. They use the concept of *benefit optimization* to design QoS control algorithms. A key design of their work is that a portion of the resources is reserved specifically to serve requests with degraded QoS. Thus, the system performs QoS negotiation and adjustment only to requests admitted into the "degraded QoS area". When a request departs, their algorithm adjusts the QoS levels of requests admitted into the degraded QoS area with the goal of maximizing the benefit. A shortcoming of their work is that it does not discuss *how much* resources should be reserved dynamically in response to changing workloads so that the system benefit is maximized at run time. Chen et al. [4] propose a utility model also with the goal of benefit maximization by managing system resources through admission control and dynamic QoS adaptation of concurrent requests. Their approach does not use the concept of designated "degraded QoS areas", so QoS adaptation can be performed on all admitted requests as long as the QoS profiles of admitted requests permit. Since system resources are not partitioned to specifically serve requests of different priority classes, low-priority requests can possibly use up all system resources, when the arrival rate of low-priority requests is much higher than that of high-priority requests. This may result in high-priority clients being rejected and consequently a lower overall benefit obtained by the system if high-priority requests carry a much higher benefit value. In this latter case, reserving resources designated to serve only high-priority clients will be more beneficial.

In this paper, we present a self-adjusting QoS control scheme also with the goal of benefit optimization. We first consider the case in which the system contains a high-priority class and a low-priority class with two service levels $Q_{\max}$ and $Q_{\min}$. Then we discuss how the scheme can be generalized to a more complex setting with higher number of service classes and larger number of service levels. We adopt the concept of a designated QoS degraded area such that QoS adaptation only applies to low-priority requests admitted into the QoS degraded area. This design is considered less intrusive and makes business sense, that is, users who pay more expect to receive a certain QoS guarantee throughout the service lifetime. A typical example is to classify clients into three groups:

(1) high-priority clients who receive a constant QoS throughout and pay more;
(2) low-priority clients who also receive a constant QoS throughout and pay less;
(3) low-priority clients who may receive varying QoS levels in the range of $[Q_{\max}, Q_{\min}]$ during the service lifetime and pay the least.

Thus, low-priority requests admitted into a QoS degraded area are the designated clients vulnerable to QoS adjustment and are fully aware that their QoS may be adjusted after being admitted; however, as a compensation they will pay the lowest fare among all service classes. Our scheme explicitly partitions system resources to serve requests in these three groups. The best partitioning is dictated by the workload and benefit/penalty characteristics of requests in difference priority classes. It can be that the best partition when given a particular workload condition is that no reserved partition exists for only serving high-priority or low-priority only requests, i.e., the "QoS degraded area" is the sole area open to all clients in which the QoS adaptation is applied to low-priority requests. Thus, our self-adjusting QoS control scheme encompasses special cases considered in the literature, including the case of no reserved, partitioned areas for different priority requests as in [4], and the case of no designated "degraded QoS area" as in [3]. The results obtained from our scheme can be used to dynamically partition system resources for admission control in response to changing client workloads so as to optimize the system performance.

2. System model

We assume that the system consists of a number of QoS slots, each of which corresponds to the minimum amount of resource reservation required to service a client with the lowest QoS requirement. For a video server, for example, the QoS requirement in a slot corresponds to the smallest frame size with black and white display. Naturally, there exists a maximum number of such QoS slots under which the system can service without overloading, as having been addressed in previous work in admission control [2,3]. Clients with higher QoS requirements must each occupy two or more such slots, e.g., for a video server, this may correspond to a bigger frame size with color video display.

We first consider a special case in which there exist two priority classes of clients, with each class being characterized by its own arrival/departure rates and reward/penalty values. The arrival rates of high-priority and low-priority clients are λ_h and λ_l, respectively, while the departure rates are μ_h and μ_l, respectively. The system ensures that customers' minimum QoS requirements are satisfied by performing admission control. We assume that a high priority client specifies a QoS requirement and once the QoS requirement is accepted by the server, it is not to be changed or renegotiated. On the other hand, a low-priority client will specify a range of QoS requirements, thus giving the system some leverage to renegotiate its QoS when necessary. The renegotiation can be done in two ways:

(1) the system can lower the QoS of low-priority clients in order to accommodate more clients into the system when the resource becomes scarce;
(2) the system can raise the QoS of low-priority clients when the resource becomes rich again.

Thus, the system can adjust the QoS level of low-priority clients based on the workload to the system, although it must maintain the same QoS level for high-priority clients. That is, the absolute QoS guarantee applies to high-priority clients while the best-effort QoS applies to low-priority clients. For ease of exposition, we again first consider a special case in which the minimum QoS requirement ($Q_{\min}$) of a low-priority client is exactly one half of its maximum QoS requirement ($Q_{\max}$) with $Q_{\max}$ being the same as that required for a high-priority client. A low-priority client thus has two QoS levels that would allow the system to do QoS control. The general assumptions of multiple priority classes and QoS levels will be treated later in Section 3.

From the perspective of the server system, the system behaves as if it contains N capacity slots. When all slots are used up, the server can lower the QoS level of low-priority clients, if any is found, to accommodate newly arriving clients, provided that doing so can improve the "pay-off" of the system. The pay-off to the server when a client completes its service is characterized by each client's reward and penalty parameters. We assume that the reward which a high-priority client brings to the system is v_h if it is served successfully; on the other hand, the reward which a low-priority client brings to the system depends on the QoS level received: it is v_l during the proportion of the time in which it is being served at the $Q_{\max}$ level and v_{ll} during the proportion of the time in which it is being served at the $Q_{\min}$ level, with $v_l \geqslant v_{ll}$. On the flip side, we assume that the penalties to the system when high-priority and low-priority clients are rejected are q_h and q_l, respectively, with $q_h \geqslant q_l$.

The performance metric being considered in the paper takes both rewards and penalties of clients into consideration. It is called the system's reward rate defined as the average amount of value received by the server per time unit. In other words, under a particular admission policy if the system on average services $\mathcal{N}_h$ high-priority clients, $\mathcal{N}_l$ low-priority clients at the maximum QoS level and $\mathcal{N}_{ll}$ low-priority clients with the minimum QoS level per unit time while it rejects $\mathcal{M}_h$ high-priority clients and $\mathcal{M}_l$ low-priority clients per unit time, then the system's average reward rate is:

$$\mathcal{N}_h v_h + \mathcal{N}_l v_l + \mathcal{N}_{ll} v_{ll} - \mathcal{M}_h q_h - \mathcal{M}_l q_l. \tag{1}$$

This reward rate can be translated into the profit rate of a company running the on-demand multimedia/telecommunication service business. The problem that we are interested in solving thus is to identify the best self-adjusting QoS control scheme under which this performance metric is maximized, as a function of model input variables, including $N, \lambda_h, \lambda_l, \mu_h, \mu_l, v_h, v_l, v_{ll}, q_h$ and q_l defined above.

3. Self-adjusting QoS control

Our self-adjusting QoS control scheme partitions the N resource slots into three parts: n_h, n_l and n_m, with n_h being specifically allocated to high-priority clients, n_l being specifically allocated to low-priority clients and the remaining n_m slots being in the designated "degraded QoS area" sharable to both types of clients. Each slot is capable of servicing a high-priority client or a low-priority client running at the $Q_{\max}$ level. The system always fills in the slots in n_h and n_l for arriving high- and low-priority clients, respectively, before filling in a slot in n_m.

In this proposed scheme, low-priority clients being admitted into the n_m part are the "designated" clients vulnerable to QoS fluctuations. When a low-priority client arrives, if there is a slot available in the n_l or n_m part, then the low-priority client is accepted; otherwise, it is rejected. Similarly, when a high-priority client arrives, if there is a slot available in the n_h or n_m part, then the client is accepted. Otherwise, the system further checks if currently there are at least 2 low-priority clients each occupying a slot in the n_m part, i.e., each being served at the $Q_{\max}$ level. If no, then the arriving high-priority client is rejected immediately. Otherwise, two such low-priority clients in the n_m part will each reduce their QoS level from $Q_{\max}$ to $Q_{\min}$, i.e., each occupying only one half of a slot, thus giving up a slot in the n_m part to accommodate the newly arriving high-priority client. Conversely, when a high-priority client or a low-priority client running at the $Q_{\max}$ level departs in the n_m part, if there exist two low-priority clients each currently running at the $Q_{\min}$ level, then the QoS level of such two low-priority clients will be increased to $Q_{\max}$, thus consuming the resource released by the departing client. If the departing client was only running at the $Q_{\min}$ level, then only one low-priority client can increase its QoS level to $Q_{\max}$, if it exists.

Note that this self-adjusting QoS control scheme encompasses the special case in which there is no space specifically reserved for low-priority clients, i.e., $n_l = 0$. In this special case, all low-priority clients are equally treated, that is, they all have to compete with high-priority clients in the n_m part and all can experience QoS fluctuations. A subcase of this special case is that both n_l and n_h are zero, in which case no space is specifically reserved for high-priority clients either.

3.1. Analysis

We derive an approximate solution of the average reward rate obtainable based on an analytical model. The solution is approximate because

(1) it does not keep track of the numbers of clients in the n_h, n_m and n_l parts globally and instead it only takes the steady state spill-over rates from the n_h and n_l parts as the arrival rates of clients into the n_m part; and
(2) it does not track the time periods in which a low-priority client in the n_m part stays at the $Q_{\max}$ and $Q_{\min}$ levels and thus it only calculates the reward rate based the QoS level of the low-priority client as the client departs.

The analytical model considers the n_h and n_l parts as $M/M/n_h/n_h$ and $M/M/n_l/n_l$ queues, respectively, and the n_m part as a Markov chain with the arrival rates of high-priority and low-priority clients into the n_m part as the "spill-over" rates from the n_h part and the n_l part, respectively. The spill-over rates of high-priority and low-priority clients are given by:

$$\Lambda_h = \lambda_h \times \frac{\frac{1}{n_h!}\left(\frac{\lambda_h}{\mu_h}\right)^{n_h}}{1+\sum_{j=1}^{n_h}\frac{1}{j!}\left(\frac{\lambda_h}{\mu_h}\right)^j}$$

and

$$\Lambda_l = \lambda_l \times \frac{\frac{1}{n_l!}\left(\frac{\lambda_l}{\mu_l}\right)^{n_l}}{1+\sum_{j=1}^{n_l}\frac{1}{j!}\left(\frac{\lambda_l}{\mu_l}\right)^j}.$$

The Markov model for the n_m part takes into consideration of arrivals and departures of both high- and low-priority clients. In addition, it also models QoS adjustment activities of low-priority clients. Let $(n_m^{ll}, n_m^{l}, n_m^{h})$ be the state representation of the Markov model where n_m^{ll} is the number of low-priority clients being served in n_m at $Q_{\min}$, n_m^{l} is the number of low-priority clients being served in n_m at $Q_{\max}$, and n_m^{h} is the number of high-priority clients in n_m, subject to the constraint that $n_m^{ll}/2 + n_m^{l} + n_m^{h} \leqslant n_m$ in any state. Fig. 1 shows a Markov chain for the case in which $n_m = 3$. Let P_j be the steady state probability that the

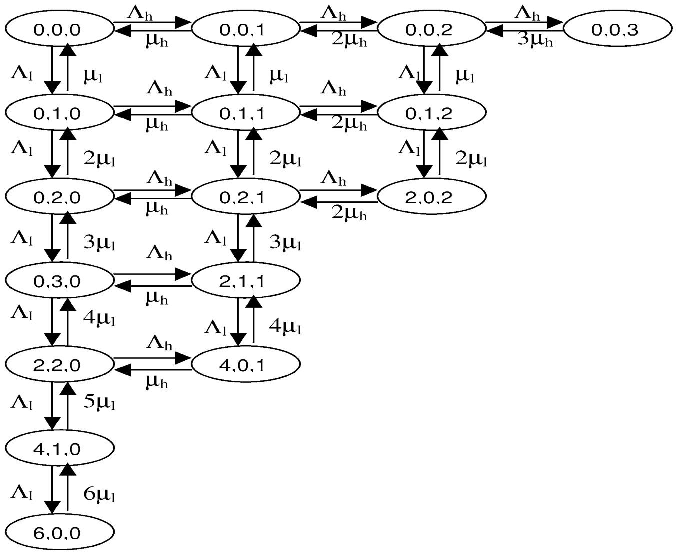

Fig. 1. Markov model for the nm part with $n_m = 3$.

n_m part is in state j. Then, based on Eq. (1), the reward rate out of the n_m part, R_{n_m}, is calculated by:

$$\begin{aligned} R_{n_m} &= \sum_{\forall j} P_j\left(n_m^{ll}\mu_l v_{ll} + n_m^{l}\mu_l v_l + n_m^{h}\mu_h v_h\right) \\ &\quad - \sum_{\forall j \text{ s.t. } n_m^{ll}/2+n_m^h=n_m} P_j q_l \Lambda_l \\ &\quad - \sum_{\substack{\forall j \text{ s.t. } n_m^{ll}/2+n_m^l+n_m^h=n_m \\ \text{and } n_m^l<2}} P_j q_h \Lambda_h, \end{aligned} \tag{2}$$

where the last two terms account for the penalties due to rejecting clients when all slots are used up in the n_m part. In Fig. 1, the system rejects low-priority clients in states (0, 0, 3), (2, 0, 2), (4, 0, 1) and (6, 0, 0) and rejects high-priority clients in these same states plus states (0, 1, 2), (2, 1, 1) and (4, 1, 0).

Combining the reward rate out of the n_m part as derived above with those from the n_h and n_l parts, we obtain the approximate solution for the total reward rate obtainable under a (n_h, n_m, n_l) partition as follows:

$$\begin{aligned} &\sum_{i=1}^{n_h} i\mu_h \times v_h \times \frac{\frac{1}{i!}\left(\frac{\lambda_h}{\mu_h}\right)^i}{1+\sum_{j=1}^{n_h}\frac{1}{j!}\left(\frac{\lambda_h}{\mu_h}\right)^j} \\ &\quad + \sum_{i=1}^{n_l} i\mu_l \times v_l \times \frac{\frac{1}{i!}\left(\frac{\lambda_l}{\mu_l}\right)^i}{1+\sum_{j=1}^{n_l}\frac{1}{j!}\left(\frac{\lambda_l}{\mu_l}\right)^j} + R_{n_m}. \end{aligned} \tag{3}$$

3.2. Multiple priority classes and QoS levels

Our scheme can be easily generalized to the case when there exist more than two priority classes. Assume that there are M priority classes, of which class 1 is the highest priority class while class M is the lowest. Class k, $1 \leqslant k \leqslant M$, is characterized by its own set of parameters $(\lambda_k, \mu_k, v_k, q_k)$. Under our scheme, $n_m + \sum_{k=1}^{M} n_k = N$ with n_m being the designated "degraded QoS area" into which all clients can be admitted but only class M clients are subject to QoS adjustment. The generalization is straightforward:

(1) the "spill-over" arrival rate Λ_k into the n_m part from clients in class k, $1 \leqslant k \leqslant M$, will be from the corresponding $M/M/n_k/n_k$ queue; and
(2) the Markov model for the n_m part will have the state representation $(n_m^{ll}, n_m^{M}, \ldots, n_m^{2}, n_m^{1})$.

The approximate solution for the reward rate obtainable is thus given by:

$$\begin{aligned} &\sum_{k=1}^{M}\left(\sum_{i=1}^{n_k} i\mu_k \times v_k \times \frac{\frac{1}{i!}\left(\frac{\lambda_k}{\mu_k}\right)^i}{1+\sum_{j=1}^{n_k}\frac{1}{j!}\left(\frac{\lambda_k}{\mu_k}\right)^j}\right) \\ &\quad + \sum_{\forall j} P_j\left(\sum_{k=1}^{M}(n_m^k\mu_k v_k) + n_m^{ll}\mu_l v_{ll}\right) \end{aligned}$$

$$- \sum_{\substack{\forall j \text{ s.t. } \sum_{k=1}^{M} n_m^k + n_m^{ll}/2 = n_m \\ \text{and } n_m^M < 2}} P_j \left(\sum_{k=1}^{M-1} q_k \Lambda_k \right)$$

$$- \sum_{\forall j \text{ s.t. } \sum_{k=1}^{M-1} n_m^k + n_m^{ll}/2 = n_m} P_j q_M \Lambda_M. \qquad (4)$$

Recall that our scheme performs QoS adjustment only to the lowest-priority clients. To generalize our scheme to multiple QoS levels where different priority clients may have different QoS requirements and the lowest-priority clients may have more than 2 QoS levels, we can consider another system parameter b_k, $1 \leqslant k \leqslant M$, such that a class k client would require b_k slots (out of the total of N slots in the system) to satisfy its (maximum) QoS requirement. We can formulate the problem such that the minimum QoS requirement ($Q_{\min}$) of the lowest-priority client in class M still requires exactly one slot and its maximum QoS requirement ($Q_{\max}$) would require b_M slots. There can be a range of QoS levels in $[1, b_M]$ for the lowest-priority clients in the n_m part for QoS adjustment, e.g., b_M, $b_M - 1$, and so on till 1. Now the problem becomes finding the best partition of $(N_1, N_2, \ldots, N_M, N_m)$ such that

$$N_m + \sum_{k=1}^{M} N_k = N \quad \text{and} \quad N_k/b_k = n_k$$

(number of class k clients in the N_k part). For the N_m part, initially a class k client (including a class M client) will need b_k slots available to be admitted. When there are not enough slots, the system can lower the QoS of existing class M clients (if exist) from b_M to $b_M - 1$ first, and subsequently from $b_M - 1$ to $b_M - 2$ and so on until the QoS level of existing class M clients all goes down to 1 in order to accommodate new arrivals, after which new arrivals will be rejected. The approximate solution can be obtained for this extension by considering M $M/M/n_k/n_k$ queues (with $n_k = N_k/b_k$) each describing the N_k part, $1 \leqslant k \leqslant M$, and a Markov model for describing the N_m part. The expression for the reward rate obtainable in this case will be similar to Eq. (4) except that we obtain the R_{n_m} term based on the new Markov model.

4. Simulation validation

The objective of the simulation study is to collect performance data so as to demonstrate the validity of our scheme and to compare the average reward obtainable by the system as a result of executing the self-adjusting QoS control scheme with those by two other schemes:

- Baseline #1: this is the "degraded QoS area" only scheme, i.e., the same self-adjusting QoS control is applied to the "degraded QoS area" part; however, n_m is the only area in the system.
- Baseline #2: this is the "partitioning" only scheme, i.e., the same partitioning of system resources is applied; however, the "degraded QoS area" although still admitting clients of all classes does not perform QoS adjustment on any class.

We study the design of a multimedia system [8] for the case $N = 32$ to illustrate the applicability of our proposed self-adjusting QoS control scheme. For a selected (n_h, n_m, n_l) value set, we compute the average reward rate obtained by the system due to the self-adjusting QoS control scheme by the *batch means* method. Under this method, the simulation program is executed for a long run divided into batches. A sample mean is computed in each batch. Using these batch means, we then compute the grand mean and the confidence interval. During a batch run, we compute the accumulated reward as a rejection or a departure occurs. A rejected high-priority (low-priority) client takes q_h (q_l, respectively) of reward away. A completed high-priority client adds v_h of reward. For a low-priority client, we keep track of the proportion of time it is being served at the $Q_{\max}$ ($Q_{\min}$) level. If a low-priority client had been served with x time units at $Q_{\max}$ and y time units at $Q_{\min}$ when it departed, then the reward added to the system would be $v_l \times x/(x+y) + v_{ll} \times y/(x+y)$. At the end of each batch run, we divide the accumulated reward by the batch run period to get the mean average reward rate for that batch run. A sufficient number of batches are run in the simulation study to make sure that the grand average reward rate obtained has an accuracy of 5% at a 95% confidence level.

Table 1 lists the optimal (n_h, n_m, n_l) value sets and reward rates obtained under the self-adjusting

Table 1
Optimizing (n_h, n_m, n_l) set under the self-adjusting QoS control scheme for $N = 32$

$(\lambda_h, \lambda_l, \mu_h, \mu_l, v_h, v_l, v_{ll}, q_h, q_l)$	Optimal (n_h, n_m, n_l)	Our scheme reward rate (analytical)	Our scheme reward rate (simulation)	Baseline #1 reward rate	Baseline #2 reward rate
(10, 20, 1, 1, 10, 5, 2, 2, 1)	(0.32, 0)	191.7	191.7	191.7	180.1
(10, 40, 1, 1, 10, 5, 2, 2, 1)	(0, 32, 0)	185.0	185.0	185.0	167.9
(10, 60, 1, 1, 10, 5, 2, 2, 1)	(1, 31, 0)	150.7	152.3	149.9	147.8
(10, 120, 1, 1, 10, 5, 2, 2, 1)	(11, 13, 8)	86.5	87.7	66.0	87.2
(20, 10, 1, 1, 10, 5, 2, 2, 1)	(0, 32, 0)	236.6	236.6	236.6	227.8
(20, 20, 1, 1, 10, 5, 2, 2, 1)	(0, 32, 0)	236.8	236.8	236.8	219.1
(20, 40, 1, 1, 10, 5, 2, 2, 1)	(19, 13, 0)	199.6	197.9	193.0	196.8
(20, 60, 1, 1, 10, 5, 2, 2, 1)	(20, 12, 0)	176.5	175.8	150.7	176.0
(20, 80, 1, 1, 10, 5, 2, 2, 1)	(21, 11, 0)	155.5	155.3	115.0	155.6
(20, 60, 1, 1, 10, 10, 2, 2, 1)	(10, 0, 22)	246.9	246.9	161.7	253.2
(20, 60, 1, 1, 10, 8, 2, 2, 1)	(16, 0, 16)	210.8	210.8	157.3	216.0
(20, 60, 1, 1, 10, 6, 2, 2, 1)	(20, 0, 12)	184.2	184.1	152.9	187.7
(20, 60, 1, 1, 10, 4, 2, 2, 1)	(21, 11, 0)	175.8	175.3	148.5	165.8
(20, 60, 1, 1, 10, 2, 1, 2, 1)	(24, 8, 0)	156.5	156.3	104.1	149.5

QoS control scheme, with respect to some selected sets of input model parameter values characterizing various client workload possibilities to the server system for $N = 32$. Under the column "our scheme reward rate" we list values collected by simulation and by calculation based on the analytical model discussed in Section 3.2 for validation. We also listed the reward rates obtained by two baseline schemes for comparison.

From Table 1, we observe two results. First, the self-adjusting QoS control scheme at the optimal point can often outperform baseline scheme 1 by a significant margin. The effect is especially pronounced when the system is heavily loaded. This result indicates that for a given set of workload conditions, there is always an optimal way of allocating resources. Moreover, it is better that we only open a *designated* degraded QoS area (i.e., n_m) to apply QoS control instead of opening up all resources (i.e., all N slots) undiscriminantly as in baseline scheme 1 so that the system can adapt to workload changes more effectively to maximize the system reward. Second, the crossover point at which the self-adjusting QoS control scheme starts to perform better than baseline scheme 2 in this case study is when $v_h + 2v_{ll} > 2_{v_l} - q_h$ is true. This is so because the left hand side represents the reward obtained by degrading the QoS level of two low-priority clients running at $Q_{\max}$ in the n_m part to accommodate an arriving high-priority client, while the right hand side represents the reward which would have been obtained had the QoS adjustment not been performed. The difference thus represents the reward gain by performing self-adjusting QoS control. Furthermore, when the above condition is violated, the system tends to allocate more resource slots to low-priority clients in the n_l part to prevent self-adjusting QoS control in the n_m part from occurring so as to optimize the reward rate. This is especially so when the arrival rate of low-priority clients is much higher than that of high-priority clients. In general, however, we do expect the condition $v_h + 2v_{ll} > 2v_l - q_h$ holds true for most systems so that the system can use the results presented in this paper to adjust the QoS of low-priority clients in the n_m part to accommodate more clients into the system and to maximize the reward rate obtainable by the system. Lastly, we should note that the values presented in Table 1 are rate parameters (e.g., reward per time unit), so a difference of 1 can be significant, e.g., in terms of dollars/second.

References

[1] S. Brandt, G. Nutt, T. Berk, M. Humphrey, Soft real-time application execution with dynamic quality of service assurance, in: 6th International Workshop on Quality of Service, Napa, CA, May 1998.

[2] E. Chang, A. Zakhor, Cost analysis for VBR video servers, in: IEEE Multimedia, Winter 1996, pp. 56–71.

[3] I.R. Chen, C.M. Chen, Threshold-based admission control policies for multimedia server, Comput. J. 39 (9) (1996) 757–766.

[4] L. Chen, S. Khan, K.F. Li, E. Manning, Building an adaptive multimedia system using the utility model, in: Lecture Notes in Comput. Sci., Vol. 1586, Springer, Berlin, 1999, pp. 289–298.

[5] W. Lee, B. Sabata, Admission control and QoS negotiation for soft-real time applications, in: IEEE International Conference on Multimedia Computing and Systems, Florence, Italy, June 1999, pp. 147–152.

[6] B. Li, K. Nahrstedt, A control theoretical model for quality of service adaptation, in: 6th International Workshop on Quality of Service, Napa, CA, May 1998.

[7] M.A. Marsan, S. Marano, C. Mastroianni, M. Meo, Performance analysis of cellular communication networks supporting multimedia services, in: ACM/Baltzer Mobile Networks and Applications, Vol. 5, 2000, pp. 167–177.

[8] H.M. Vin, A. Goyal, P. Goyal, Algorithms for designing multimedia servers, Comput. Comm. 18 (3) (1995) 192–203.