ELSEVIER

Computer Networks 35 (2001) 307–326

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

Admission control schemes to provide class-level QoS in multiservice networks ☆

Suresh Kalyanasundaram [a,1], Edwin K.P. Chong [b], Ness B. Shroff [b,*]

[a] *Networks and Infrastructure Research Labs, Motorola Labs, Schaumburg, IL 60196, USA*

[b] *School of Electrical and Computer Engineering, Purdue University, 1285 Electrical Engineering Building, West Lafayette, IN 47907-1285, USA*

Received 7 April 2000; accepted 19 July 2000

Responsible Editor: G. Pacifici

Abstract

In this work, we consider the scenario where different classes of traffic with differing bandwidth requirements are to be supported over a link with finite capacity. The network will be required to meet class-level quality-of-service (QoS) constraints that are specified as relations between the call blocking probabilities of these traffic classes. The network will also be required to maintain maximum throughput performance. We develop two different admission control schemes that consider these two important criteria. First, we choose an objective that maximizes the throughput subject to the constraint that the maximum deviation from the desired class-level QoS is smaller than a certain pre-specified quantity. In the second approach, we choose an objective that minimizes the deviation from the desired class-level QoS subject to the throughput being greater than a pre-specified minimum value. We show how these problems can be converted to linear programming problems. We illustrate our schemes using numerical examples.

Keywords: Quality of service; Call admission control; Class-level QoS; Constrained Markov decision process; Linear program

1. Introduction

The trend of future networks, whether wired or wireless, is towards multiservice networks, where a single network infrastructure will support a variety of applications such as data, voice, and video [1]. Thus future networks will handle traffic belonging to different classes that have different quality-of-service (QoS) and

☆ This research was supported in part by the National Science Foundation through grants NCR-9624525, CDA-9617388, ECS-9501652, and ANI-9805441, and by DARPA/ITO through grant F19628-98-C-0051.

* Corresponding author. Tel.: +1-765-494-3471; fax: +1-765-494-3358.

E-mail addresses: kalyanas@rsch.comm.mot.com (S. Kalyanasundaram), echong@ecn.purdue.edu (E.K.P. Chong), shroff@ecn.purdue.edu (N.B. Shroff).

[1] This work was done when the author was a graduate student at Purdue University.

1389-1286/01/$ - see front matter
PII: S1389-1286(00)00173-0

bandwidth requirements. In this work, we are concerned with the call blocking probability requirement of different classes of users. The scenario we consider is one where calls belonging to several traffic classes compete for bandwidth on a link with finite capacity. All calls belonging to one traffic class are assumed to require the same amount of bandwidth. We also assume that users that are not admitted immediately are lost, i.e., we do not assume call queueing.

The assumption that each user requires a certain amount of bandwidth is clearly applicable to circuit-switched networks, e.g., circuit-switched wireless and optical networks. However, it does not restrict us to the case of circuit-switched networks alone. For packet-switched networks, although there is no physical reservation of bandwidth for each user, a logical notion of bandwidth requirement is still essential to provide QoS [2]. In packet-switched networks with statistical QoS guarantees, the amount of bandwidth needed by a user would be given by the effective or equivalent bandwidth [3–5]. The above view of abstracting the bandwidth or rate requirement of a user hides the packet-level details, and simplifies the analysis. A justification for such a decoupling can be found in [6].

When a new call arrives, a *call admission control* decision occurs in which a controller decides whether or not the call can be admitted based on the current "state" of the system and the overall objectives of the network. The call admission control problem has been well-studied under different contexts. A common call admission policy is the *complete sharing* policy [7–10]. Under this scheme, all the available link capacity is open to all classes of users. The admission controller, under the complete sharing policy, determines if there is enough capacity to admit a newly arriving call and admits or rejects the call depending on whether or not there is enough capacity. This scheme is very simple, but does not meet the call blocking probability expectations of different call classes. Therefore, different classes will have widely differing call blocking probabilities. For example, in Section 2, we derive the call blocking probability for the complete sharing scheme and show that it does not have the flexibility to control the call blocking probabilities of call classes.

The *complete partitioning* [8–10] scheme, on the other hand, divides the entire capacity into as many partitions as there are classes of users, and a new user that finds insufficient capacity in the partition for its class is dropped. The complete partitioning scheme can be tuned to control the relationship between the call blocking probabilities of various classes of users, but the resulting throughput may be very low. A hybrid of the complete sharing and complete partitioning policies is the *partial sharing* policy. In the partial sharing policy, each call class has a dedicated portion of bandwidth reserved for it, and all the classes compete for the remaining unreserved portion of the bandwidth [11].

Other admission control schemes have been developed that meet specific objectives. For example, in [8,12], the authors formulate the problem of finding an admission control strategy that maximizes the throughput as a continuous-time Markov decision process. The authors of [13] are interested in making all classes of users experience the same call blocking probability. They use a game-theoretic approach to obtain the admission control algorithm.

Several future developments are likely to require call admission schemes to meet *class-level QoS* requirements. By class-level QoS, we mean the desired relations between the call blocking probabilities of different call classes. It is foreseen that appropriate pricing mechanisms will be introduced to prevent congestion in future networks [14]. The network will offer various levels of service with appropriate prices associated with them. All users who pay identical amounts for the same application belong to the same class. Clearly, users who pay more should be accorded priority and should belong to a higher priority class. The call blocking probability of high priority classes should be smaller than that of low priority classes. Such a requirement on the call blocking probability of call classes is an example of a class-level QoS requirement. Another example of a class-level QoS requirement is "fair blocking", where it is desired that all call classes experience the same call blocking probability. We have a very general formulation of class-level QoS requirements that includes the above examples as special cases. In our work, we consider relations of a special kind that achieve "weighted-fair blocking". (Detailed descriptions of fair blocking and weighted-fair

blocking are given in Section 2.) Besides class-level QoS, throughput efficiency is another important factor to consider while designing call admission schemes. In this work, we design call admission schemes that consider these two criteria.

The rest of the paper is organized as follows. In Section 2, we illustrate how call blocking probabilities of different call classes may be obtained by considering the complete sharing scheme and motivate the need for more sophisticated call admission schemes to provide class-level QoS. We take two different approaches that naturally combine the two criteria of throughput-efficiency and achieving the desired class-level QoS. First, we choose an objective that maximizes the throughput subject to a constraint on the maximum deviation from the desired relations between the call blocking probabilities of various call classes. This approach is discussed in Section 3. In Section 4, our objective is to minimize the deviation from the desired class-level QoS subject to a minimum throughput constraint. Numerical results comparing the performance of our schemes with the "throughput-maximizing" scheme and the complete sharing scheme are presented in Section 5.

2. Call blocking probabilities under the complete sharing scheme

As described in the introduction, the complete sharing scheme admits calls as long as there is enough unused capacity in the system. In this section, we derive the call blocking probabilities experienced by different call classes under the complete sharing scheme and motivate the need for more sophisticated call admission schemes. Let the number of call classes be M. The bandwidth requirement of calls in each class is fixed, but the requirement may vary from class to class. Let the bandwidth requirement of a class-i user be e_i bandwidth units (BWUs), and let the capacity of the link be C BWUs. Let calls of class i, $i = 1, 2, \ldots, M$, arrive according to an independent Poisson process with rate λ_i calls/s, and let the call holding duration of class-i calls be independent and exponentially distributed with mean $1/\mu_i$ s. As stated earlier, we do not assume call queueing. Therefore, the decision to admit or reject a call has to be made immediately upon call arrival. As mentioned earlier, the assumption that each class-i user requires e_i BWUs is very general and can be used in both circuit-switched and packet-switched networks.

We denote the state of the system by a vector of M components $(n_1, n_2, \ldots, n_M)$ such that n_i represents the number of active class-i users in the system. Let $\{\boldsymbol{X}(t);\ t \geqslant 0\}$ represent the state of the system as it evolves over time. It then follows that $\boldsymbol{X}(t)$ is a continuous-time Markov chain under the complete sharing policy. We define the set S of all feasible states as

$$S = \left\{ \boldsymbol{n} = (n_1, n_2, \ldots, n_M) : \sum_{i=1}^{M} n_i e_i \leqslant C \text{ and } n_i \in \mathbb{Z}^+ \text{ for } i = 1, 2, \ldots, M \right\},$$

where $\mathbb{Z}^+$ is the set of all non-negative integers. The set S includes all possible states such that the combined bandwidth requirement of the active users does not exceed the link bandwidth.

For ease of description, we define $\boldsymbol{\delta}_i$ to be a vector of M components with zeros in all except the ith position where it has a one. The transition rate from state $\boldsymbol{n}$ to state $\boldsymbol{p}$, denoted by $q_{(\boldsymbol{n},\boldsymbol{p})}$, is given as follows:

$$q_{(\boldsymbol{n},\boldsymbol{p})} = \begin{cases} \lambda_i & \text{if } \boldsymbol{p} = \boldsymbol{n} + \boldsymbol{\delta}_i \text{ and } \boldsymbol{p} \in S, \\ n_i \mu_i & \text{if } \boldsymbol{p} = \boldsymbol{n} - \boldsymbol{\delta}_i \text{ and } \boldsymbol{p} \in S, \\ 0 & \text{otherwise,} \end{cases}$$

where n_i denotes the ith component of the vector $\boldsymbol{n}$. Note that the transition from state $\boldsymbol{n}$ to state $\boldsymbol{n} + \boldsymbol{\delta}_i$ is made with a rate that is equal to the call arrival rate of class-i calls if $\boldsymbol{n} + \boldsymbol{\delta}_i \in S$. This follows from the complete sharing policy of admitting calls as long as there is enough idle capacity to do so. In

general, the value of the transition rate from state $\boldsymbol{n}$ to state $\boldsymbol{n}+\boldsymbol{\delta}_i$ depends on the choice of the admission control algorithm. The admission control algorithms described in the following sections use different criteria for call admission, and hence these transition rates get modified appropriately. However, the transition rate from $\boldsymbol{n}$ to $\boldsymbol{n}-\boldsymbol{\delta}_i$ will remain the same irrespective of the chosen admission control policy. Let $\pi_{\boldsymbol{n}}$ denote the steady-state probability of the system being in state $\boldsymbol{n}$. Then the balance equations are as follows:

$$\pi_{\boldsymbol{n}}\left(\sum_{\{i:\boldsymbol{n}+\boldsymbol{\delta}_i\in S\}}\lambda_i+\sum_{i=1}^{M}n_i\mu_i\right)=\sum_{\{i:\boldsymbol{n}+\boldsymbol{\delta}_i\in S\}}\pi_{\boldsymbol{n}+\boldsymbol{\delta}_i}(n_i+1)\mu_i+\sum_{\{i:n_i\neq 0\}}\pi_{\boldsymbol{n}-\boldsymbol{\delta}_i}\lambda_i \quad \text{for all } \boldsymbol{n}\in S.$$

On the left-hand side of the above equation is the rate of leaving state $\boldsymbol{n}$, and on the right-hand side is the rate of entering state $\boldsymbol{n}$. Additionally, the $\pi_{\boldsymbol{n}}$ have to satisfy the following normalization condition:

$$\sum_{\{\boldsymbol{n}:\boldsymbol{n}\in S\}}\pi_{\boldsymbol{n}}=1.$$

Note that $\pi_{\boldsymbol{n}}$ can be interpreted as the fraction of time the system spends in state $\boldsymbol{n}$. It then follows from the Poisson arrivals see time averages (PASTA) property that the call blocking probability for class-i calls is

$$B_i=\sum_{\{\boldsymbol{n}:\boldsymbol{n}\in S \text{ and } \boldsymbol{n}+\boldsymbol{\delta}_i\notin S\}}\pi_{\boldsymbol{n}},$$

where the right-hand side is just the fraction of time the system is in a state that does not admit class-i calls.

Using the above method we can determine the call blocking probabilities of various call classes. In Fig. 1, we plot the call blocking probability obtained for the case where there are two classes of traffic. Here, we assume that the call arrival rate and the call holding duration of both traffic classes are identical. The call arrival rate of each call class is 0.01 calls/s, and the average call holding duration is 2 min. The

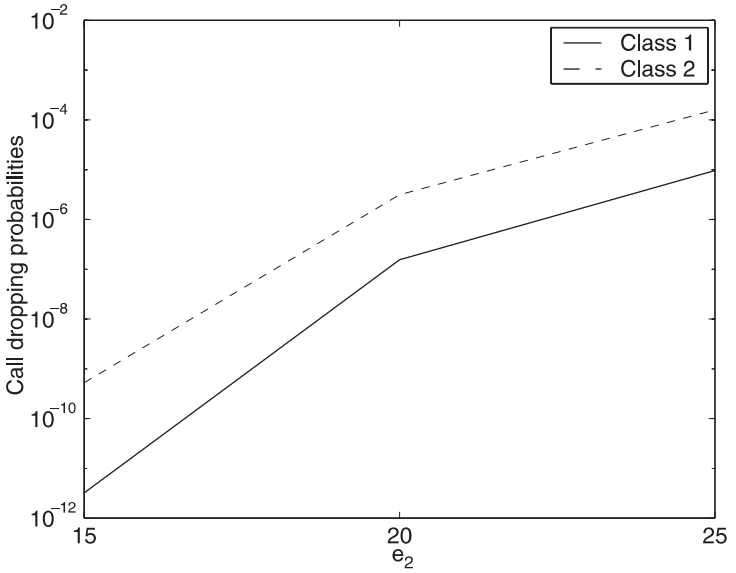


Fig. 1. Comparison of the call blocking probability obtained for two different classes of users that have differing bandwidth requirements.

two classes of traffic differ only in their bandwidth requirement. The value of e_2 is used as a parameter in the figure, and we fix $e_1 = 1$ BWU, and $C = 200$ BWUs. The difference between the call blocking probabilities of the two classes is about two orders of magnitude. Clearly, the complete sharing policy does not provide flexibility in controlling the call blocking probabilities of classes. As discussed in the introduction, admission control schemes need to have the flexibility to control the call blocking probabilities of various call classes.

Depending on the characteristics of the call classes, their call blocking probabilities will be expected to satisfy several conditions. We call the desired relations between the call blocking probabilities of call classes as the desired class-level QoS requirement. For example, in fair blocking the objective is to ensure that the difference in call blocking probabilities of any two different call classes is small. In other words, the desired relation among the call blocking probabilities of call classes is $B_1 = B_2 = \cdots = B_M$, where B_i denotes the call blocking probability of class-i users. In other situations, it would be necessary to give preferential treatment to high-priority call classes. To handle all such situations, we consider the general case of "weighted-fair blocking". (Although we consider the case of weighted-fair blocking in this work, our methods apply in the general setting of any linear relation involving the call blocking probabilities. We restrict ourselves to the case of weighted-fair blocking for ease of exposition.) In weighted-fair blocking, the objective is to reduce the difference between appropriately scaled call blocking probabilities. Thus, the desired relation between call blocking probabilities of classes is given by $w_{ij}B_i = w_{ji}B_j$, $1 \leqslant i, j \leqslant M$, $i \neq j$ for appropriately chosen positive weights w_{ij}. A requirement that class-1 users experience half the call blocking probability of class-2 users is an example of weighted fair blocking, where we give preferential treatment to class-1 users. We note that there need not exist desired relations between the call blocking probabilities of every pair of classes, which is a special case of what we are considering. Another special case of our weighted-fair blocking is the case of bounds on the maximum call blocking probability of each call class (see Section 3). Also the weights w_{ij} need to be chosen so that they do not result in inconsistent requirements on the B_i. If the chosen w_{ij} results in inconsistent requirements on the B_i, our method will reveal that it is so.

While providing class-level QoS is important, the network service provider also has to maintain good throughput performance. The problem of maximizing throughput has been studied by other authors (e.g., [8,12]), but there has been little work that considers the two problems of maximizing throughput and providing class-level QoS together. Our objective is to develop admission control schemes that consider the two criteria of meeting class-level QoS requirements and maintaining adequate throughput performance. In the following two sections, we take two different approaches to this problem.

3. Maximizing throughput subject to a class-level QoS constraint

In this section, we describe a call admission scheme that maximizes throughput subject to a constraint on the maximum deviation from the desired class-level QoS relations. As in the previous section, we assume that class-i calls arrive according to an independent Poisson process of rate λ_i and that the call holding durations of class-i calls are independent and exponentially distributed with mean $1/\mu_i$. There are M classes of users with each class-i user requiring e_i BWUs. The set of all allowed states S is as defined in Section 2.

Given the current system state, the admission controller makes an admission control decision on the classes of calls that will be admitted in that state. This decision would depend on the state of the system and could potentially depend on the time at which the decision is taken, and the entire history of the system. In each state, the admission controller can choose one out of several possible actions. Formally, an action is denoted by a vector $\boldsymbol{u}$ such that $\boldsymbol{u} = (u_1, u_2, \ldots, u_M)$ with $u_i = 0$ or 1. If $u_i = 0$ then the admission controller

blocks any call belonging to class i, and if $u_i = 1$ it accepts any call belonging to class i. We define the set $K_{\boldsymbol{n}}$ of all allowed control actions in state $\boldsymbol{n}$ as follows:

$$K_{\boldsymbol{n}} = \{(u_1, u_2, \ldots, u_M) : u_i = 0 \text{ or } 1 \text{ for } i = 1, 2, \ldots, M \text{ and } \boldsymbol{n} + \boldsymbol{\delta}_i u_i \in S \text{ for } i = 1, 2, \ldots, M\}.$$

Note that an action $\boldsymbol{u}$ is allowed in state $\boldsymbol{n}$ only if there is enough idle capacity in the system to admit a call belonging to any of the traffic classes that the action $\boldsymbol{u}$ decides to admit. The set $K_{\boldsymbol{n}}$ contains precisely these actions. For each state $\boldsymbol{n} \in S$, the set $K_{\boldsymbol{n}}$ has at most 2^M elements. A *control policy* is a sequence of such admission control actions made in each state as the system evolves. We let $\mathscr{F}'$ denote the set of all such control policies. Our objective is to obtain an admission control policy that optimizes the following problem:

$$\underset{f \in \mathscr{F}'}{\text{maximize}} \quad T(f) \tag{1}$$

$$\text{subject to} \quad \max\{|w_{ls}B_l(f) - w_{sl}B_s(f)| : 1 \leqslant l < M,\ l+1 \leqslant s \leqslant M\} \leqslant \epsilon,$$

where $0 < \epsilon < 1$ is a pre-specified constant, and $T(f)$ is the throughput (time-average of bandwidth usage) obtained using the policy f. The value of ϵ can be set to be the maximum deviation from the desired class-level QoS that can be tolerated. Clearly, the throughput and the call blocking probability of each call class depends on the chosen control policy $f \in \mathscr{F}'$. We show this explicitly in the problem formulation above. The constraint in the problem in Eq. (1) prevents a deviation larger than the allowed tolerance of ϵ from the desired relation between call blocking probabilities. As mentioned earlier, we consider the case of weighted-fair blocking, where the desired relation between call blocking probabilities is $w_{ls}B_l = w_{sl}B_s$ for $1 \leqslant l, s \leqslant M$, $l \neq s$, and we obtain the case of fair blocking by setting $w_{ls} = 1$ for all l and s. We rewrite the constraint in the problem in Eq. (1) as the following $M(M-1)$ constraints:

$$w_{ls}B_l - w_{sl}B_s \leqslant \epsilon \quad \text{for } 1 \leqslant l,\ s \leqslant M \text{ and } l \neq s. \tag{2}$$

From the above equation, it follows that the case where each call class has a bound on its maximum call blocking probability can be obtained by the following slight modification. We merely let $w_{sl} = 0$ and have a separate ϵ for each call class corresponding to the desired maximum call blocking probability of that class.

We will now define a per-unit-time reward such that the long-run average reward per unit time represents the throughput obtained from the chosen control policy $f \in \mathscr{F}'$. When the system stays in state $\boldsymbol{n} = (n_1, n_2, \ldots, n_M)$ and when decision $\boldsymbol{u}$ is chosen, reward is accrued at the rate of

$$r_{\boldsymbol{n}}^{\boldsymbol{u}} = \sum_{i=1}^{M} n_i e_i \quad \text{for all } \boldsymbol{u} \in K_{\boldsymbol{n}}, \tag{3}$$

where $K_{\boldsymbol{n}}$ is the set of all allowed control actions in state $\boldsymbol{n}$. We note that the reward in state $\boldsymbol{n}$ is independent of the chosen action in that state. The reward $r_{\boldsymbol{n}}^{\boldsymbol{u}}$ in state $\boldsymbol{n}$ is the amount of bandwidth in use in that state. From the definition of Eq. (3) it immediately follows that the long-run expected reward per unit time is the throughput of the system. Given that we start in state $\boldsymbol{m}$, the following definition of the long-run average reward corresponds to the throughput obtained from the chosen policy $f \in \mathscr{F}'$:

$$R_{\boldsymbol{m}}(f) = \lim_{t \to \infty} E_f\left[\frac{Z(t)}{t} \,\middle|\, \boldsymbol{X}(0) = \boldsymbol{m}\right], \tag{4}$$

where $\boldsymbol{X}(t)$ is the state of the system at time t, $Z(t)$ the total reward accrued until time t, and $E_f(\cdot)$ is the expectation operator taken with respect to policy f. If the limit does not exist in the above equation, we agree to take the lim inf instead.

Besides the reward $r_{\boldsymbol{n}}^{\boldsymbol{u}}$, we will now associate costs with each state $\boldsymbol{n}$ and for each action $\boldsymbol{u} \in K_{\boldsymbol{n}}$ such that the long-run average cost is the term on the left-hand side of Eq. (2). (We distinguish between cost and reward depending on whether we want to minimize or maximize the long-run average of this quantity.) To do this, note that we can obtain the term B_l as the long-run average, if we associate a per-unit-time cost of one with states that choose an action that rejects class-l calls and a per-unit-time cost of zero with states that choose an action that admits class-l calls. The long-run average of this cost definition is the proportion of time the system spends in a state that rejects class-l calls. From this observation, it follows that the following per-unit-time cost definition will yield the term $w_{ls}B_l - w_{sl}B_s$ as the long-run average:

$$c_{\boldsymbol{n}}^{\boldsymbol{u}}(l,s) = \begin{cases} 0 & \text{if } \boldsymbol{u}_l = \boldsymbol{u}_s = 1, \\ w_{ls} & \text{if } \boldsymbol{u}_l = 0 \text{ and } \boldsymbol{u}_s = 1, \\ -w_{sl} & \text{if } \boldsymbol{u}_l = 1 \text{ and } \boldsymbol{u}_s = 0, \\ w_{ls} - w_{sl} & \text{if } \boldsymbol{u}_l = \boldsymbol{u}_s = 0. \end{cases} \tag{5}$$

Therefore, if the term $Z(t)$ in Eq. (4) is the cumulative cost up to time t obtained from the costs $c_{\boldsymbol{n}}^{\boldsymbol{u}}(l,s)$ instead of the rewards $r_{\boldsymbol{n}}^{\boldsymbol{u}}$, then Eq. (4) represents the term $w_{ls}B_l - w_{sl}B_s$. Eq. (5) also reveals that the cost definition can be appropriately extended to any linear relation involving the call blocking probabilities.

If the system is in state $\boldsymbol{n}$ and action $\boldsymbol{u} \in K_{\boldsymbol{n}}$ is chosen, then the next state of the system is determined according to the rates $q_{(\boldsymbol{n},\boldsymbol{p})}^{\boldsymbol{u}}$, and these rates depend only on the current state of the system $\boldsymbol{n}$ and the chosen control action $\boldsymbol{u}$. The transition rates $q_{(\boldsymbol{n},\boldsymbol{p})}^{\boldsymbol{u}}$ are obtained as follows:

$$q_{(\boldsymbol{n},\boldsymbol{p})}^{\boldsymbol{u}} = \begin{cases} \lambda_i & \text{if } \boldsymbol{p} = \boldsymbol{n} + \boldsymbol{\delta}_i \text{ and } \boldsymbol{u}_i = 1, \\ n_i\mu_i & \text{if } \boldsymbol{p} = \boldsymbol{n} - \boldsymbol{\delta}_i \text{ and } n_i > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Note that the control action $\boldsymbol{u}$ does not have any effect on call termination. Also note that the transition rates $q_{(\boldsymbol{n},\boldsymbol{p})}^{\boldsymbol{u}}$, costs $c_{\boldsymbol{n}}^{\boldsymbol{u}}(l,s)$, and rewards $r_{\boldsymbol{n}}^{\boldsymbol{u}}$ are functions only of the last state and the subsequent action.

In summary, once the system reaches state $\boldsymbol{n}$ and action $\boldsymbol{u}$ is chosen, the following occur:

- A per-unit-time reward of $r_{\boldsymbol{n}}^{\boldsymbol{u}}$ is accrued for the entire duration of stay in state $\boldsymbol{n}$.
- A per-unit-time cost of $c_{\boldsymbol{n}}^{\boldsymbol{u}}(l,s)$ is incurred, one for each term $w_{ls}B_l - w_{sl}B_s$, where $1 \leqslant l,s \leqslant M$ and $l \neq s$.
- The next state is determined according to the transition rate $q_{(\boldsymbol{n},\boldsymbol{p})}^{\boldsymbol{u}}$ given in Eq. (6).

From the above discussion it follows that we have a constrained continuous-time Markov decision problem to solve (see [15–17]). We will now show that we can search for the optimal policy over a smaller set instead of the original set $\mathscr{F}'$. First, note that the set $\mathscr{F}'$ has control policies that depend on the time at which the admission control decision is made. For example, it includes the control policy that specifies that during the first visit to state $\boldsymbol{m}$ a specific control action has to be chosen, but starting from the second visit an alternate control action has to be chosen. Such control policies that depend on the time at which the decision is made are called *non-stationary policies*. Control policies that do not depend on the time of decision making are called *stationary policies*. An important subclass of stationary policies are *pure policies* that choose the same decision $\boldsymbol{u}$ during every visit to a state. Note that the class of stationary policies also includes *randomized policies* that choose an action according to the same probability distribution during every visit to a state. It is known (see [16,18]) that if every pure policy f gives rise to a single recurrent class plus a set (possibly empty) of transient states, and if there exists a policy that satisfies the constraints (in Eq. (2)), then there exists an optimal stationary policy. In our problem, every pure policy does give rise to a Markov chain with a single recurrent class. To see this, observe that the state $(0,0,\ldots,0)$ is accessible from all states for every pure policy. Therefore, that state must be part of every recurrent class for each pure policy. But recurrent classes are disjoint. Therefore, we have just a single recurrent class for every pure policy. Using the above result, we can now restrict our search for optimal policies among the set of all stationary policies $\mathscr{F}$. It has

also been shown that under the above conditions, the optimal value of $R_{\boldsymbol{m}}$ does not depend on the initial state $\boldsymbol{m}$ [19].

To characterize the set $\mathscr{F}$ of all stationary policies we denote the probability that we choose control action $\boldsymbol{u} \in K_{\boldsymbol{n}}$ when the system is in state $\boldsymbol{n} \in S$ as $d_{\boldsymbol{n}}^{\boldsymbol{u}}$. The $d_{\boldsymbol{n}}^{\boldsymbol{u}}$ for $\boldsymbol{u} \in K_{\boldsymbol{n}}$ and $\boldsymbol{n} \in S$ are the decision variables of the problem. It is clear that

$$\sum_{\boldsymbol{u} \in K_{\boldsymbol{n}}} d_{\boldsymbol{n}}^{\boldsymbol{u}} = 1 \quad \text{for } \boldsymbol{n} \in S \tag{7}$$

and that

$$d_{\boldsymbol{n}}^{\boldsymbol{u}} \geqslant 0 \quad \text{for } \boldsymbol{u} \in K_{\boldsymbol{n}} \text{ and } \boldsymbol{n} \in S. \tag{8}$$

A stationary control policy is the following set of probability distributions, one for each feasible state:

$$\left\{ d_{\boldsymbol{n}}^{\boldsymbol{u}} : \boldsymbol{u} \in K_{\boldsymbol{n}} \text{ and } \boldsymbol{n} \in S \right\}. \tag{9}$$

Implicit in the above notation is that the control policy does not depend on the time of decision making. In other words, successive visits to a particular state will employ the same probability distribution to choose the control action. The set $\mathscr{F}$ consists of distributions of the form given in Eq. (9) satisfying Eqs. (7) and (8). Based on the above definition of a stationary control policy, we have the following definitions for pure and randomized policies. A stationary control policy is called a *pure policy* if for each $\boldsymbol{n} \in S$ there is an action $\boldsymbol{u} \in K_{\boldsymbol{n}}$ such that $d_{\boldsymbol{n}}^{\boldsymbol{u}} = 1$. A stationary policy that is not pure is a *randomized policy*.

Under any given randomized policy f, the states the system visits form a continuous-time Markov chain with transition rates given as follows:

$$q_{(\boldsymbol{n},\boldsymbol{p})}(f) = \sum_{\{\boldsymbol{u}:\boldsymbol{u} \in K_{\boldsymbol{n}}\}} q_{(\boldsymbol{n},\boldsymbol{p})}^{\boldsymbol{u}} d_{\boldsymbol{n}}^{\boldsymbol{u}}. \tag{10}$$

We let $\pi_{\boldsymbol{n}}(f)$ be the limiting steady-state probability of the continuous-time Markov chain for the stationary policy f. We know that these probabilities have to satisfy the following conditions:

$$\pi_{\boldsymbol{n}}(f) \sum_{\boldsymbol{p} \in S} \sum_{\boldsymbol{u} \in K_{\boldsymbol{n}}} q_{(\boldsymbol{n},\boldsymbol{p})}^{\boldsymbol{u}} d_{\boldsymbol{n}}^{\boldsymbol{u}} - \sum_{\boldsymbol{i} \in S} \sum_{\boldsymbol{u} \in K_{\boldsymbol{i}}} \pi_{\boldsymbol{i}}(f) q_{(\boldsymbol{i},\boldsymbol{n})}^{\boldsymbol{u}} d_{\boldsymbol{i}}^{\boldsymbol{u}} = 0 \quad \text{for } \boldsymbol{n} \in S, \tag{11}$$

$$\begin{aligned} &\sum_{\boldsymbol{n} \in S} \pi_{\boldsymbol{n}}(f) = 1, \\ &\pi_{\boldsymbol{n}}(f) \geqslant 0 \quad \text{for } \boldsymbol{n} \in S. \end{aligned} \tag{12}$$

Eq. (11) is the balance equation for the continuous-time Markov chain and the interpretation in Eq. (10) is useful in seeing that Eq. (11) is valid. With the above definition of $\pi_{\boldsymbol{n}}(f)$, the long-run average reward of Eq. (4) reduces to the following equation and is independent of the initial state $\boldsymbol{m}$ [19]:

$$\sum_{\boldsymbol{n} \in S} \sum_{\boldsymbol{u} \in K_{\boldsymbol{n}}} \pi_{\boldsymbol{n}}(f) r_{\boldsymbol{n}}^{\boldsymbol{u}} d_{\boldsymbol{n}}^{\boldsymbol{u}}.$$

The product of $\pi_{\boldsymbol{n}}(f)$ and $d_{\boldsymbol{n}}^{\boldsymbol{u}}$ is the fraction of time the system is in state $\boldsymbol{n}$ and action $\boldsymbol{u}$ is chosen. The above equation is merely the weighted average reward with weights being the product of $\pi_{\boldsymbol{n}}(f)$ and $d_{\boldsymbol{n}}^{\boldsymbol{u}}$. Similarly, the constraints on $w_{ls}B_l - w_{sl}B_s$ can be written as follows:

$$\sum_{\boldsymbol{n} \in S} \sum_{\boldsymbol{u} \in K_{\boldsymbol{n}}} \pi_{\boldsymbol{n}}(f) c_{\boldsymbol{n}}^{\boldsymbol{u}}(l,s) d_{\boldsymbol{n}}^{\boldsymbol{u}} \leqslant \epsilon \quad \text{for } 1 \leqslant l,\ s \leqslant M,\ \ l \neq s.$$

Thus we have the following optimization problem to solve:

$$\begin{aligned}
\text{maximize} \quad & \sum_{n\in S}\sum_{u\in K_n}\pi_n(f)r_n^u d_n^u \\
\text{subject to} \quad & \pi_n(f)\sum_{p\in S}\sum_{u\in K_n}q_{(n,p)}^u d_n^u - \sum_{i\in S}\sum_{u\in K_i}\pi_i(f)q_{(i,n)}^u d_i^u = 0 \quad \text{for } n\in S, \\
& \sum_{n\in S}\pi_n(f) = 1, \\
& \pi_n(f) \geqslant 0 \quad \text{for } n\in S, \\
& \sum_{u\in K_n} d_n^u = 1 \quad \text{for } n\in S, \\
& d_n^u \geqslant 0 \quad \text{for } u\in K_n \text{ and } n\in S, \\
& \sum_{n\in S}\sum_{u\in K_n}\pi_n(f)c_n^u(l,s)d_n^u \leqslant \epsilon \quad \text{for } 1\leqslant l,\ s\leqslant M,\ \ l\neq s.
\end{aligned}$$

Setting $x_n^u = \pi_n(f)d_n^u \geqslant 0$ for $n\in S$ and $u\in K_n$ and using the fact that $\pi_n(f) = \sum_{u\in K_n} x_n^u$ for $n\in S$, we have

$$\begin{aligned}
\text{maximize} \quad & \sum_{n\in S}\sum_{u\in K_n} r_n^u x_n^u \\
\text{subject to} \quad & \sum_{p\in S}\sum_{u\in K_n} x_n^u q_{(n,p)}^u - \sum_{i\in S}\sum_{u\in K_i} q_{(i,n)}^u x_i^u = 0 \quad \text{for } n\in S, \qquad (13) \\
& \sum_{n\in S}\sum_{u\in K_n} x_n^u = 1, \\
& x_n^u \geqslant 0 \quad \text{for } n\in S, \\
& \sum_{n\in S}\sum_{u\in K_n} c_n^u(l,s)x_n^u \leqslant \epsilon \quad \text{for } 1\leqslant l,\ s\leqslant M,\ \ l\neq s. \qquad (14)
\end{aligned}$$

One of the equations among the set of constraints in Eq. (13) can be written in terms of the other equations. Therefore, we can drop one of the constraints in Eq. (13). The above optimization problem is a linear program and can be solved using well-known techniques, like the simplex algorithm. The simplex algorithm will reveal if the conditions in Eq. (14) are too stringent. Policy iteration and value iteration techniques [20] for solving Markov decision processes cannot be applied to solve this problem due to the additional constraints in Eq. (14).

To obtain the optimal d_n^{u*} of the problem, we note that

$$d_n^u = x_n^u \Big/ \sum_{u\in K_n} x_n^u$$

Therefore, once the x_n^{u*} have been computed, the optimal control action in each state is obtained according to the following procedure. Let S^* be the set

$$S^* = \left\{ n\in S : x_n^{u*} > 0 \text{ for some } u\in K_n \right\}. \qquad (15)$$

Then the optimal action in state n is determined as follows. For $n\in S^*$, action v is chosen with probability $x_n^{v*}/\sum_{u\in K_n} x_n^{u*}$, and for $n\notin S^*$, an arbitrary action $v\in K_n$ is chosen with probability one. Each time the system visits a state $n\in S^*$ that has more than one action with non-zero probability, an action is chosen according to the probability distribution d_n^{u*}.

It has been shown in [16] that for a Markov decision process with k additional constraints (apart from the ones that a valid steady-state probability distribution has to satisfy), if the above procedure is used to

obtain the optimal policy, then there is a list of at most $|S| + k$ actions from which the optimal policy is chosen. This implies that the optimal policy obtained by the above procedure has at most k states that require randomization. The number of additional constraints in our problem is $M(M-1)/2$ because half the constraints in Eq. (14) are automatically satisfied if the other half are satisfied. Therefore, the optimal policy obtained by the above procedure requires randomization in at most $M(M-1)/2$ states, and the numerical results in Section 5 agree with this result. Our example in Section 5 also shows that the admission control scheme designed using the above procedure ensures that the deviation from the desired class-level QoS does not exceed the threshold ϵ.

4. Providing class-level QoS subject to a minimum throughput constraint

In this section, we develop an admission control algorithm that minimizes the deviation from the desired class-level QoS subject to a constraint on the throughput. The reference scenario is the same as described in the last section. Our objective is to obtain an admission control policy that is the solution to the following optimization problem:

$$\min_{f \in \mathscr{F}'} \quad \max\{|w_{ls}B_l(f) - w_{sl}B_s(f)| : 1 \leqslant l < M,\ l+1 \leqslant s \leqslant M\} \tag{16}$$

$$\text{subject to} \quad T(f) \geqslant T_{\min},$$

where $T(f)$ denotes the throughput of policy f, and $T_{\min}$ is the minimum throughput that we want the admission control policy to achieve. In the above problem, we explicitly show the dependence of the call blocking probability and the throughput on the admission control policy f. As in the last section, we are justified in restricting our search for optimal policies among stationary policies $\mathscr{F}$. We now rewrite the objective in Eq. (16) as

$$\min_{f \in \mathscr{F}} \quad \max\{w_{ls}B_l - w_{sl}B_s : 1 \leqslant l \leqslant M,\ 1 \leqslant s \leqslant M,\ l \neq s\}, \tag{17}$$

where we drop the absolute value signs by taking the maximum over $M(M-1)$ quantities.

The above problem can be formulated as a continuous-time Markov decision process with constraints. The state space, action space, and transition rates of the problem are the same as those described in the last section. We now identify the (per-unit-time) cost for staying in state $\boldsymbol{n} \in S$ under control action $\boldsymbol{u} \in K_{\boldsymbol{n}}$. We define a vector of costs for staying in state $\boldsymbol{n}$ under control action $\boldsymbol{u} \in K_{\boldsymbol{n}}$ to obtain each of the $M(M-1)$ terms in Eq. (17) as the long-run average. We define our vector in such a way that the long-run average of the first component yields $w_{12}B_1 - w_{21}B_2$, the long-run average of the second component will yields $w_{13}B_1 - w_{31}B_3$, and so on. Such a cost vector $\boldsymbol{c}_{\boldsymbol{n}}^{\boldsymbol{u}}$ is determined according to the following algorithm.

Algorithm (*Determination of cost vector*).

(i) Initialize $i = 1$.
(ii) For $l = 1, 2, \ldots, M$ do the following:
(iii) For $s = 1, 2, \ldots, M$ and $s \neq l$ do the following:
(iv) Set $c_{\boldsymbol{n}}^{\boldsymbol{u}}(i) = c_{\boldsymbol{n}}^{\boldsymbol{u}}(l, s)$ according to Eq. (5) and $i = i + 1$.

The cost vector, as can be seen from the above algorithm, has $M(M-1)$ components with one term each for the $M(M-1)$ terms in the objective function in Eq. (17). A control action that accepts class-1 calls but rejects class-2 calls contributes $-w_{21}$ to $w_{12}B_1 - w_{21}B_2$, while a control action that accepts both class-1 and class-2 calls does not contribute anything to $w_{12}B_1 - w_{21}B_2$. Using this definition of cost vector, the term $w_{12}B_1 - w_{21}B_2$ can be written as

$$\sum_{\boldsymbol{n}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{n}}}\pi_{\boldsymbol{n}}(f)c_{\boldsymbol{n}}^{\boldsymbol{u}}(1)d_{\boldsymbol{n}}^{\boldsymbol{u}},$$

where the $\pi_{\boldsymbol{n}}(f)$, the $d_{\boldsymbol{n}}^{\boldsymbol{u}}$, etc., are as defined in the last section. To meet the constraint on the throughput, we use the same reward definition as in Eq. (3). The minimum throughput constraint can now be written as

$$\sum_{\boldsymbol{n}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{n}}}\pi_{\boldsymbol{n}}(f)r_{\boldsymbol{n}}^{\boldsymbol{u}}d_{\boldsymbol{n}}^{\boldsymbol{u}}\geqslant T_{\min}.$$

The value of $T_{\min}$ has to be chosen appropriately to prevent the problem from becoming infeasible. For example, if the value of $T_{\min}$ is too high, then even the admission control policy that maximizes throughput (without any constraints) will not achieve $T_{\min}$. In the numerical results, we choose $T_{\min}$ to be αT_{cs}, where $\alpha<1$, and T_{cs} is the throughput achieved by the complete sharing scheme.

We thus have the following optimization problem to solve:

$$\min_{f\in\mathscr{F}}\quad \max\left\{\sum_{\boldsymbol{n}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{n}}}\pi_{\boldsymbol{n}}(f)c_{\boldsymbol{n}}^{\boldsymbol{u}}(l)d_{\boldsymbol{n}}^{\boldsymbol{u}}:1\leqslant l\leqslant M(M-1)\right\} \tag{18}$$

$$\begin{aligned}
\text{subject to}\quad & \pi_{\boldsymbol{n}}(f)\sum_{\boldsymbol{p}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{n}}}q_{(\boldsymbol{n},\boldsymbol{p})}d_{\boldsymbol{n}}^{\boldsymbol{u}}-\sum_{\boldsymbol{i}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{i}}}\pi_{\boldsymbol{i}}(f)q_{(\boldsymbol{i},\boldsymbol{n})}^{\boldsymbol{u}}d_{\boldsymbol{i}}^{\boldsymbol{u}}=0 \quad \text{for } \boldsymbol{n}\in S,\\
& \sum_{\boldsymbol{n}\in S}\pi_{\boldsymbol{n}}(f)=1,\\
& \pi_{\boldsymbol{n}}(f)\geqslant 0 \quad \text{for } \boldsymbol{n}\in S,\\
& \sum_{\boldsymbol{u}\in K_{\boldsymbol{n}}}d_{\boldsymbol{n}}^{\boldsymbol{u}}=1 \quad \text{for } \boldsymbol{n}\in S,\\
& d_{\boldsymbol{n}}^{\boldsymbol{u}}\geqslant 0 \quad \text{for } \boldsymbol{u}\in K_{\boldsymbol{n}} \text{ and } \boldsymbol{n}\in S,\\
& \sum_{\boldsymbol{n}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{n}}}\pi_{\boldsymbol{n}}(f)r_{\boldsymbol{n}}^{\boldsymbol{u}}d_{\boldsymbol{n}}^{\boldsymbol{u}}\geqslant T_{\min}.
\end{aligned}$$

We now show that the above optimization problem can be converted to $M(M-1)$ linear programming problems. Our approach makes use of the following result, the proof of which is given in Appendix A.

Lemma 1. *Let $\boldsymbol{x}^*$ be a minimizer of the following problem*:

$$\begin{aligned}
&\min_{\boldsymbol{x}}\quad \max\{g_1(\boldsymbol{x}),g_2(\boldsymbol{x})\}\\
&\text{subject to}\quad \boldsymbol{x}\in\Omega,
\end{aligned} \tag{19}$$

and let $\boldsymbol{y}_1^*$ and $\boldsymbol{y}_2^*$ be minimizers, if they exist, of the following two problems, respectively:

$$\begin{aligned}
&\text{minimize}\quad g_1(\boldsymbol{x})\\
&\text{subject to}\quad \boldsymbol{x}\in\Omega,\\
&\qquad\qquad g_1(\boldsymbol{x})\geqslant g_2(\boldsymbol{x}),
\end{aligned} \tag{20}$$

and

$$\begin{aligned}
&\text{minimize}\quad g_2(\boldsymbol{x})\\
&\text{subject to}\quad \boldsymbol{x}\in\Omega,\\
&\qquad\qquad g_2(\boldsymbol{x})\geqslant g_1(\boldsymbol{x}).
\end{aligned} \tag{21}$$

Then we have that

$$\min\{g_1(\boldsymbol{y}_1^*), g_2(\boldsymbol{y}_2^*)\} = \max\{g_1(\boldsymbol{x}^*), g_2(\boldsymbol{x}^*)\} \tag{22}$$

with the understanding that if either $\boldsymbol{y}_1^*$ *or* $\boldsymbol{y}_2^*$ *does not exist, then the corresponding term on the left-hand side of Eq.* (22) *is taken to be* ∞.

Using the above lemma and by successively considering one of the $M(M-1)$ terms in Eq. (18) as the maximum among all the $M(M-1)$ terms, we can convert our minimax problem into $M(M-1)$ optimization problems. We consider one such optimization problem and show how it can be converted to a linear program. We consider the following optimization problem:

$$\begin{aligned}
\text{minimize} \quad & \sum_{\boldsymbol{n}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{n}}} \pi_{\boldsymbol{n}}(f)c_{\boldsymbol{n}}^{\boldsymbol{u}}(1)d_{\boldsymbol{n}}^{\boldsymbol{u}} \\
\text{subject to} \quad & \pi_{\boldsymbol{n}}(f)\sum_{\boldsymbol{p}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{n}}} q_{(\boldsymbol{n},\boldsymbol{p})}d_{\boldsymbol{n}}^{\boldsymbol{u}} - \sum_{\boldsymbol{i}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{i}}} \pi_{\boldsymbol{i}}(f)q_{(\boldsymbol{i},\boldsymbol{n})}^{\boldsymbol{u}}d_{\boldsymbol{i}}^{\boldsymbol{u}} = 0 \quad \text{for } \boldsymbol{n}\in S, \\
& \sum_{\boldsymbol{n}\in S}\pi_{\boldsymbol{n}}(f) = 1, \\
& \pi_{\boldsymbol{n}}(f) \geqslant 0 \quad \text{for } \boldsymbol{n}\in S, \\
& \sum_{\boldsymbol{u}\in K_{\boldsymbol{n}}} d_{\boldsymbol{n}}^{\boldsymbol{u}} = 1 \quad \text{for } \boldsymbol{n}\in S, \\
& d_{\boldsymbol{n}}^{\boldsymbol{u}} \geqslant 0 \quad \text{for } \boldsymbol{u}\in K_{\boldsymbol{n}} \text{ and } \boldsymbol{n}\in S, \\
& \sum_{\boldsymbol{n}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{n}}} \pi_{\boldsymbol{n}}(f)r_{\boldsymbol{n}}^{\boldsymbol{u}}d_{\boldsymbol{n}}^{\boldsymbol{u}} \geqslant T_{\min}, \\
& \sum_{\boldsymbol{n}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{n}}} \pi_{\boldsymbol{n}}(f)(c_{\boldsymbol{n}}^{\boldsymbol{u}}(1) - c_{\boldsymbol{n}}^{\boldsymbol{u}}(j))d_{\boldsymbol{n}}^{\boldsymbol{u}} \geqslant 0 \quad \text{for } j = 2,3,\ldots,M(M-1),
\end{aligned}$$

where the additional constraints have been obtained by using Lemma 1. Essentially, the above problem minimizes the term $w_{12}B_1 - w_{21}B_2$ subject to the condition that it is the largest among all the $M(M-1)$ terms in Eq. (18).

Setting $x_{\boldsymbol{n}}^{\boldsymbol{u}} = \pi_{\boldsymbol{n}}(f)d_{\boldsymbol{n}}^{\boldsymbol{u}} \geqslant 0$ for $\boldsymbol{n}\in S$ and $\boldsymbol{u}\in K_{\boldsymbol{n}}$ as before, we have the following linear program to solve:

$$\begin{aligned}
\text{minimize} \quad & \sum_{\boldsymbol{n}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{n}}} x_{\boldsymbol{n}}^{\boldsymbol{u}}c_{\boldsymbol{n}}^{\boldsymbol{u}}(1) \\
\text{subject to} \quad & \sum_{\boldsymbol{p}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{n}}} x_{\boldsymbol{n}}^{\boldsymbol{u}}q_{(\boldsymbol{n},\boldsymbol{p})}^{\boldsymbol{u}} - \sum_{\boldsymbol{i}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{i}}} q_{(\boldsymbol{i},\boldsymbol{n})}^{\boldsymbol{u}}x_{\boldsymbol{i}}^{\boldsymbol{u}} = 0 \quad \text{for } \boldsymbol{n}\in S, \\
& \sum_{\boldsymbol{n}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{n}}} x_{\boldsymbol{n}}^{\boldsymbol{u}} = 1, \\
& x_{\boldsymbol{n}}^{\boldsymbol{u}} \geqslant 0 \quad \text{for } \boldsymbol{n}\in S, \\
& \sum_{\boldsymbol{n}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{n}}} x_{\boldsymbol{n}}^{\boldsymbol{u}}r_{\boldsymbol{n}}^{\boldsymbol{u}} \geqslant T_{\min}, \\
& \sum_{\boldsymbol{n}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{n}}} x_{\boldsymbol{n}}^{\boldsymbol{u}}(c_{\boldsymbol{n}}^{\boldsymbol{u}}(1) - c_{\boldsymbol{n}}^{\boldsymbol{u}}(j)) \geqslant 0 \quad \text{for } j = 2,\ldots,M(M-1).
\end{aligned} \tag{23}$$

We solve $M(M-1)$ such linear programs and the solution corresponding to the problem that yields the least objective function value is the optimal $x_{\boldsymbol{n}}^{\boldsymbol{u}*}$. Once the optimal $x_{\boldsymbol{n}}^{\boldsymbol{u}*}$ has been obtained, the optimal policy is obtained in the same manner as described in the last section.

The number of constraints in our problem is $|S| + M(M-1)$. But $M(M-1)/2 - 1$ of the constraints in (23) are automatically satisfied if the others in (23) are satisfied. Therefore, there is a list of at most $|S| + M(M-1)/2 + 1$ actions from which the optimal policy is chosen. Consequently, the optimal policy requires randomization in at most $M(M-1)/2 + 1$ states. The numerical results shown in Section 5 agree with this observation.

An important feature of the algorithms described in Sections 3 and 4 is that they do not require real-time computation of the admission control decision. The decisions may be computed beforehand and a table look-up can be done when needed. Besides reducing computational complexity, the table look-up procedure also reduces the delay before an admission decision is made. However, the numerical results in Section 5 also show that the optimal policy depends on the parameters of the system, namely the call arrival rates and call holding durations of all call classes. Therefore, our admission control decisions should be updated as needed to adapt to changing traffic conditions. An important issue is determining how often and when the admission control decision needs to be recomputed. We present some alternatives here.

(i) *Periodic updates*: The admission control decision can be recomputed after periodic intervals of time.

(ii) *Absolute change-based updates*: When the measured deviation from the desired class-level QoS exceeds a certain threshold, we can initiate a decision update.

(iii) *Relative change-based updates*: When the measured deviation from the desired class-level QoS changes by more than a certain percentage since the last update, we can initiate an update of the admission control decision.

Because the admission controller has to take decisions without complete knowledge and under uncertainty, it would be interesting to study its impact on the admission control decision. Given a range of possible values for the call arrival rates and the call holding durations, the methods described in [21–23] can be used to perform sensitivity analysis of the chosen policy and to obtain max-min and max-max optimal policies. Sensitivity analysis of a policy is performed to obtain the range of values that the average per-unit-time reward can take given the range of values that the call arrival rates and the call holding durations take. Max-min optimal policy, on the other hand, is a policy that maximizes the worst-case average reward per unit time.

5. Numerical results

In this section, we present numerical results and compare the performance of our admission control algorithms with that of the throughput maximizing and the complete sharing admission control schemes. For the numerical results, we assume that there are two classes of calls and that the total capacity of the channel is 4 BWUs. We also assume that class-1 calls require 1 BWU and class-2 calls require 2 BWUs, i.e., $e_1 = 1$ BWU, and $e_2 = 2$ BWUs. As before, calls arrive according to a Poisson process with λ_1 and λ_2 being the call arrival rates of class-1 and class-2 calls, respectively. The call holding durations for both classes of calls are assumed exponential with mean $1/\mu = 60$ s. We fix the call arrival rate of class-2 calls at 0.5 calls/s and treat λ_1 as a parameter.

In Table 1, we show the probabilities with which we should choose actions in a given state to maximize throughput. As stated earlier, this problem has been tackled before in [8,12]. Nevertheless, we give the numerical results for our particular case here to facilitate comparison with our other schemes. We show the results for three different values of λ_1. We put an '$\times$' for the probabilities of actions in "don't care" states. For example, in the maximization of throughput example in Table 1, for $\lambda_1 = 0.05$, we find that state (1,0) is a "don't care" state. Because the optimal policy is to accept only class-2 calls in all states, state (1,0) is a transient state. Note that the optimal policy that maximizes throughput may be obtained from the method in Section 3 by dropping the constraints in Eq. (14). For the complete sharing policy, the control action in a given state $\boldsymbol{n}$ does not depend on the parameters of the system, i.e., λ_1, λ_2, μ_1, and μ_2, but depends only on

Table 1
$d_{\boldsymbol{n}}^{\boldsymbol{u}}$ values for the scheme that maximizes throughput

		$\lambda_1 = 0.05$	$\lambda_1 = 1$	$\lambda_1 = 0.44$
State (0,0)	Action (1,1)	0	1	1
	Action (1,0)	0	0	0
	Action (0,1)	1	0	0
	Action (0,0)	0	0	0
State (0,1)	Action (1,1)	0	1	0
	Action (1,0)	0	0	0
	Action (0,1)	1	0	1
	Action (0,0)	0	0	0
State (0,2)	Action (0,0)	1	1	1
State (1,0)	Action (1,1)	×	1	1
	Action (1,0)	×	0	0
	Action (0,1)	×	0	0
	Action (0,0)	×	0	0
State (1,1)	Action (1,0)	×	1	1
	Action (0,0)	×	0	0
State (2,0)	Action (1,1)	×	1	1
	Action (1,0)	×	0	0
	Action (0,1)	×	0	0
	Action (0,0)	×	0	0
State (2,1)	Action (0,0)	×	1	1
State (3,0)	Action (1,0)	×	1	1
	Action (0,0)	×	0	0
State (4,0)	Action (0,0)	×	1	1

the state $\boldsymbol{n}$. The $d_{\boldsymbol{n}}^{\boldsymbol{u}}$ for the complete sharing policy (two class case, $e_1 < e_2$) in state $\boldsymbol{n} = (n_1, n_2)$ can be written in a compact manner as $d_{\boldsymbol{n}}^{\boldsymbol{u}} = 1$, where

$$\boldsymbol{u} = \begin{cases} (1,1) & \text{if } n_1 e_1 + (n_2+1)e_2 \leqslant C, \\ (1,0) & \text{if } n_1 e_1 + (n_2+1)e_2 > C \text{ and } (n_1+1)e_1 + n_2 e_2 \leqslant C, \\ (0,0) & \text{otherwise.} \end{cases}$$

In Table 2, we show the optimal policy for the admission control problem that maximizes throughput subject to the class-level QoS constraint that the absolute difference between the call blocking probabilities of the two call classes be smaller than 10^{-2}. This scheme was discussed in Section 3. We set the weights $w_{12} = w_{21} = 1$, which corresponds to the case of fair blocking. The optimal policy for the admission control scheme described in Section 4 is given in Table 3. The parameter $T_{\min}$ is chosen to be 95% of the throughput obtained by using the complete sharing policy. The parameters (λ_1, λ_2, and μ) assumed in Tables 2 and 3 are the same as that in Table 1.

The results in the tables are intuitively appealing. For example, in Table 1, for very low values of λ_1, the optimal policy is to accept only class-2 calls. This can be explained as follows. Since the call arrival rate of class-2 calls is very high compared to the call arrival rate of class-1 calls, we might as well wait a little longer for a class-2 call and increase the throughput rather than accept a class-1 call. For $\lambda_1 = 1$, the optimal

Table 2
d_n^u values for the scheme that maximizes throughput subject to the fairness constraint with $\epsilon = 0.01$

		$\lambda_1 = 0.05$	$\lambda_1 = 1$	$\lambda_1 = 0.44$
State (0,0)	Action (1,1)	1	1	1
	Action (1,0)	0	0	0
	Action (0,1)	0	0	0
	Action (0,0)	0	0	0
State (0,1)	Action (1,1)	0.1946	0	0
	Action (1,0)	0	0	0
	Action (0,1)	0.8054	1	1
	Action (0,0)	0	0	0
State (0,2)	Action (0,0)	1	1	1
State (1,0)	Action (1,1)	1	1	1
	Action (1,0)	0	0	0
	Action (0,1)	0	0	0
	Action (0,0)	0	0	0
State (1,1)	Action (1,0)	1	1	1
	Action (0,0)	0	0	0
State (2,0)	Action (1,1)	1	0.273	0.6564
	Action (1,0)	0	0	0
	Action (0,1)	0	0.727	0.3436
	Action (0,0)	0	0	0
State (2,1)	Action (0,0)	1	1	1
State (3,0)	Action (1,0)	1	1	1
	Action (0,0)	0	0	0
State (4,0)	Action (0,0)	1	1	1

policy is the complete sharing scheme while for $\lambda_1 = 0.44$, the optimal policy admits only class-2 calls in state (0,1) and admits all classes of calls in other states. Additionally, we find that the results satisfy the claim on the number of states that require randomization. For example, in Table 1, we obtain a pure policy as the optimal policy, and in Table 2, the optimal policy requires randomization in just one state.

In Fig. 2, we plot the throughput obtained for the various schemes. The constraint on the minimax admission control policy (described in Section 4) is such that $T_{\min}$ is chosen to be 99% of the throughput obtained with the complete sharing policy. As expected, we find that the scheme that maximizes throughput performs the best. However, the better performance comes at the cost of a large difference in the call blocking probabilities of the two call classes, as seen in Fig. 3. We plot the difference in call blocking probabilities for the schemes under identical conditions in Fig. 3. We see that our fairness constraint ($|B_2 - B_1| \leqslant \epsilon$) ensures that the difference between the call blocking probabilities does not exceed the threshold we set with only a moderate sacrifice in the throughput. The values of $|B_2 - B_1|$ for the scheme described in Section 4 (minimax policy) are many orders of magnitude smaller than the scales shown in the figure. We observe that the throughput obtained for the scheme that maximizes throughput is a constant until λ_1 reaches a value of 0.44. This can be explained by noting from Table 1 that the optimal admission control policy is to admit only class-2 calls until the value of λ_1 reaches about 0.44 and that the value of λ_2 is kept a constant for the results in Fig. 2. We also note that beyond a λ_1 value of about 0.44 the performance of the complete sharing scheme and the scheme that maximizes throughput coincide.

Table 3
d_n^u values for the scheme that provides weighted fair blocking subject to the constraint that the throughput is at least 95% of that obtained by using the complete sharing scheme

		$\lambda_1 = 0.05$	$\lambda_1 = 1$	$\lambda_1 = 0.44$
State (0,0)	Action (1,1)	0	0.0008	0
	Action (1,0)	0	0	0
	Action (0,1)	1	0.9992	1
	Action (0,0)	0	0	0
State (0,1)	Action (1,1)	0	0	0
	Action (1,0)	0.0811	0	0.0002
	Action (0,1)	0.2638	1	0.2926
	Action (0,0)	0.6551	0	0.7072
State (0,2)	Action (0,0)	1	1	1
State (1,0)	Action (1,1)	0	0	0
	Action (1,0)	0	1	1
	Action (0,1)	1	0	0
	Action (0,0)	0	0	0
State (1,1)	Action (1,0)	1	0	0
	Action (0,0)	0	1	1
State (2,0)	Action (1,1)	0	0	0
	Action (1,0)	0	1	1
	Action (0,1)	1	0	0
	Action (0,0)	0	0	0
State (2,1)	Action (0,0)	1	1	1
State (3,0)	Action (1,0)	×	1	1
	Action (0,0)	×	0	0
State (4,0)	Action (0,0)	×	1	1

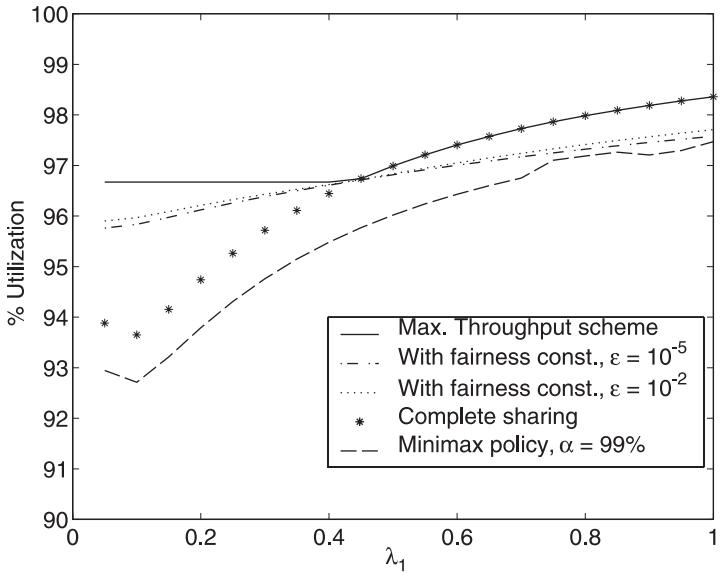


Fig. 2. The throughput obtained using various schemes.

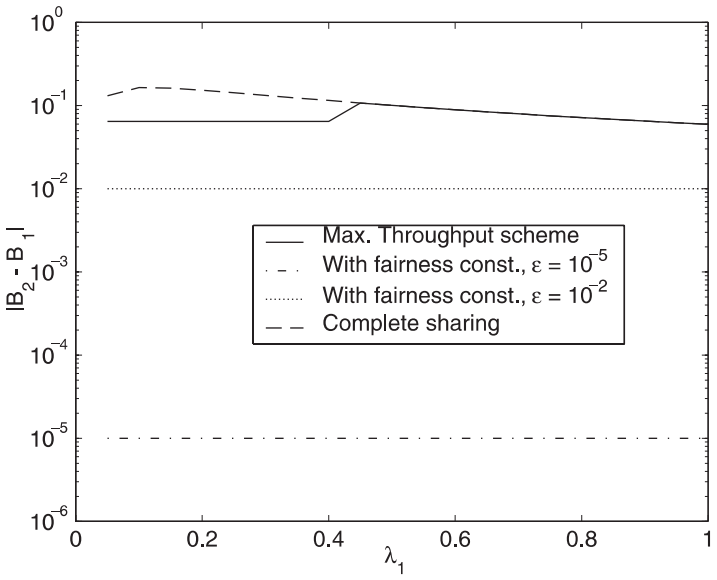


Fig. 3. Comparison of $|B_2 - B_1|$ obtained for various schemes.

6. Conclusions

In this work, we addressed the problem of providing class-level QoS to classes of users that have differing bandwidth requirements. We formulated two different problems that consider the two important criteria of providing class-level QoS and improving throughput together. One was the problem of maximizing throughput subject to the constraint that the maximum deviation from the desired class-level QoS be smaller than a pre-specified threshold. The other was the problem of minimizing the deviation from the desired class-level QoS relations subject to the constraint that the throughput achieved does not fall below a certain minimum threshold. We converted the first problem into a linear program and the second to $M(M-1)$ linear programs, where M is the number of call classes. The numerical results show that we can achieve a significant performance improvement in the class-level QoS provided by the network for only a moderate loss in throughput.

In our work, we assumed that call arrivals are Poisson and that call holding durations are exponentially distributed. While exponential call holding durations are justified for traditional voice users, there is growing evidence that it may be unsuited for other types of applications [24]. Our work here is a first step towards a solution to the optimal call admission problem for users with more general call holding time distributions. We believe that our work can be extended to more general call holding distributions. Our motivation is previous work that shows the insensitivity of the steady state distribution of the number of calls belonging to each call class to the call holding time distribution for coordinate convex admission control policies [7,11]. Thus, if the optimal admission control policy obtained by our method turns out to be coordinate convex, then that policy will continue to be optimal among all coordinate convex policies even for general call holding distributions with the same mean as the original exponential call holding durations. *Our numerical results show that the optimal policy is coordinate convex in most cases. Therefore, our policy will continue to be optimal among all coordinate convex policies in all such cases.* However, we cannot guarantee the optimality of our admission control policy among all admission control policies when the call holding durations are generally distributed. To obtain the optimal policy among all policies with general call holding distributions, more research is needed on Markov regenerative decision processes [25]. The

usefulness of such processes stems from the fact that, when call holding durations are generally distributed, the Markov property still holds at those instants when the system becomes empty. Note, however, that the Markov property no longer holds at call arrival and call termination instants.

While our scheme can be applied to network scenarios including a large number of classes and high capacity values, computing the optimal solution in such cases may involve solving MDPs with large state and action spaces. Model reduction techniques [26,27] can be used to reduce the computational complexity involved in solving MDPs with large state and action spaces. Furthermore, the solution obtained from our linear program formulation for small and medium networks can be used to design heuristic techniques that perform well in practice.

While we assumed maximizing throughput as an important objective, these ideas are easily extended to maximizing "utility". In such a modified scenario, the reward due to admitting a user would not be the amount of bandwidth needed by the user, but rather the utility that the user obtains from that bandwidth allocation. Such a modification will reflect the relative importance of bandwidth to various users, especially in the context of multiservice networks where different applications derive different amounts of benefit from the same amount of bandwidth allocation.

Appendix A. Proof of Lemma 1

Observe that $\boldsymbol{x}^*$ has to be feasible for either the problem given in Eq. (20) or the problem in Eq. (21). Therefore, it is clear that

$$\min\left\{g_1(\boldsymbol{y}_1^*), g_2(\boldsymbol{y}_2^*)\right\} \leqslant \max\left\{g_1(\boldsymbol{x}^*), g_2(\boldsymbol{x}^*)\right\}. \tag{A.1}$$

Next observe that $\boldsymbol{y}_1^*$ and $\boldsymbol{y}_2^*$, if they exist, are feasible for the problem in Eq. (19). Therefore, we have

$$\max\left\{g_1(\boldsymbol{x}^*), g_2(\boldsymbol{x}^*)\right\} \leqslant \max\{g_1(\boldsymbol{y}_1^*), g_2(\boldsymbol{y}_1^*)\}, \tag{A.2}$$

$$\max\left\{g_1(\boldsymbol{x}^*), g_2(\boldsymbol{x}^*)\right\} \leqslant \max\left\{g_1(\boldsymbol{y}_2^*), g_2(\boldsymbol{y}_2^*)\right\} \tag{A.3}$$

with the right-hand side assumed to be ∞ in Eqs. (A.2) and (A.3) if $\boldsymbol{y}_1^*$ and $\boldsymbol{y}_2^*$ do not exist, respectively. Note that at least one of them has to exist for, otherwise, $\boldsymbol{x}^*$ will not exist. Now observing that

$$\max\left\{g_1(\boldsymbol{y}_1^*), g_2(\boldsymbol{y}_1^*)\right\} = g_1(\boldsymbol{y}_1^*),$$

$$\max\left\{g_1(\boldsymbol{y}_2^*), g_2(\boldsymbol{y}_2^*)\right\} = g_2(\boldsymbol{y}_2^*)$$

we have the following two relations from Eqs. (A.2) and (A.3):

$$\max\left\{g_1(\boldsymbol{x}^*), g_2(\boldsymbol{x}^*)\right\} \leqslant g_1(\boldsymbol{y}_1^*),$$

$$\max\left\{g_1(\boldsymbol{x}^*), g_2(\boldsymbol{x}^*)\right\} \leqslant g_2(\boldsymbol{y}_2^*).$$

The above two equations taken together imply

$$\max\left\{g_1(\boldsymbol{x}^*), g_2(\boldsymbol{x}^*)\right\} \leqslant \min\left\{g_1(\boldsymbol{y}_1^*), g_2(\boldsymbol{y}_2^*)\right\}. \tag{A.4}$$

From Eqs. (A.1) and (A.4), we obtain

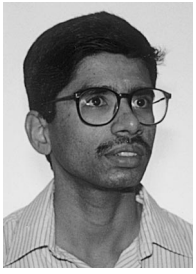
$$\max\{g_1(\boldsymbol{x}^*), g_2(\boldsymbol{x}^*)\} = \min\{g_1(\boldsymbol{y}_1^*), g_2(\boldsymbol{y}_2^*)\}$$

and the proof is complete. □

References

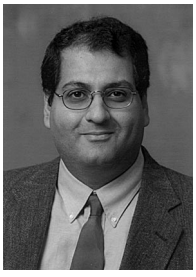
[1] J. Liebeherr, Multimedia networks: issues and challenges, Computer 28 (4) (1995) 68–69.
[2] D. Mitra, J.A. Morrison, K.G. Ramakrishnan, ATM network design and optimization: a multirate loss network framework, IEEE/ACM Transactions on Networking 4 (4) (1996) 531–543, also available from http://cm.bell-labs.com/cm/ms/who/mitra/pub.html.
[3] C.-S. Chang, J.A. Thomas, Effective bandwidth in high speed digital networks, IEEE Journal on Selected Areas in Communications 13 (1995) 1091–1100.
[4] G. Kesidis, J. Walrand, C.-S. Chang, Effective bandwidths for multiclass Markov fluids and other ATM sources, IEEE Transactions on Networking 1 (4) (1993) 424–428.
[5] E.W. Knightly, N.B. Shroff, Admission control for statistical QoS: theory and practice, IEEE Network 13 (2) (1999) 20–29, also available from http://www.ece.rice.edu/networks/publications.html.
[6] G. de Veciana, C. Courcoubetis, J. Walrand, Decoupling bandwidths: a decomposition approach to resource management in networks, in: Proceedings of the IEEE INFOCOM '94, vol. 2, Toronto, Canada, 1994, pp. 446–474, also available from http://walrandpc.eecs.berkeley.edu/Pubs.html.
[7] J.S. Kaufman, Blocking in a shared resource environment, IEEE Transactions on Communications 29 (10) (1981) 1474–1481.
[8] D. Mitra, M.I. Rieman, J. Wang, Robust dynamic admission control for unified cell and call QoS in statistical multiplexers, IEEE Journal on Selected Areas in Communications 16 (5) (1998) 692–707, also available from http://cm.bell-labs.com/cm/ms/who/mitra/pub.html.
[9] S.C. Borst, D. Mitra, Virtual partitioning for robust resource sharing: computational techniques for heterogeneous traffic, IEEE Journal on Selected Areas in Communications 16 (5) (1998) 668–678.
[10] B. Kraimeche, M. Schwartz, Circuit access control strategies in integrated digital networks, in: Proceedings of the IEEE INFOCOM '84, San Francisco, CA, 1984, pp. 230–235.
[11] J.-F.P. Labourdette, G.W. Hart, Blocking probabilities in multitraffic loss systems: insensitivity, asymptotic behavior, and approximations, IEEE Transactions on Communications 40 (8) (1992) 1355–1366.
[12] J.M. Hyman, A.A. Lazar, G. Pacifici, A separation principle between scheduling and admission control for broadband switching, IEEE Journal on Selected Areas in Communications 11 (4) (1993) 605–616.
[13] Z. Dziong, L.G. Mason, Fair-efficient call admission control policies for broadband networks – a game theoretic framework, IEEE/ACM Transactions on Networking 4 (1) (1996) 123–136.
[14] R.J. Gibbens, F.P. Kelly, Resource pricing and the evolution of congestion control, Automatica 35 (12) (1999) 1969–1985, also available from http://www.statslab.cam.ac.uk/~frank/PAPERS.
[15] S.M. Ross, Applied Probability Models with Optimization Applications, Holden-Day, San Francisco, CA, 1970.
[16] K.W. Ross, Randomized and past-dependent policies for Markov decision processes with multiple constraints, Operations Research 37 (3) (1989) 474–477.
[17] S.M. Ross, Introduction to Stochastic Dynamic Programming, Academic Press, New York, 1983.
[18] C. Derman, Finite State Markovian Decision Processes, Academic Press, New York, 1970.
[19] H. Mine, S. Osaki, Markovian Decision Processes, Elsevier, Amsterdam, 1970.
[20] M.L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, Wiley, New York, 1994.
[21] R. Givan, S. Leach, T. Dean, Bounded parameter Markov decision processes, in: Proceedings of the Fourth European Conference on Planning, 1997, pp. 234–246, also available from http://dynamo.ecn.purdue.edu/~givan.
[22] J.K. Satia, R.E. Lave Jr., Markovian decision processes with uncertain transition probabilities, Operations Research 21 (1973) 728–740.
[23] S. Kalyanasundaram, E.K.P. Chong, N.B. Shroff, Markov decision processes with uncertain transition rates: sensitivity and robustness, Tech. Rep., Purdue University, 1999.
[24] Y. Fang, I. Chlamtac, Y.-B. Lin, Modeling PCS networks under general call holding time and residence time distributions, IEEE/ACM Transactions on Networking 5 (6) (1997) 893–905.
[25] A. Pfening, M. Telek, Optimal control of Markov regenerative processes, in: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, vol. 1, 1998, pp. 663–668.
[26] T. Dean, R. Givan, S. Leach, Model reduction techniques for computing approximately optimal solutions for Markov decision processes, in: Proceedings of the Uncertainty in AI, 1997, also available from http://dynamo.ecn.purdue.edu/~givan.

[27] T. Dean, R. Givan, Model minimization in Markov decision processes, in: Proceedings of the National Conference on Artificial Intelligence, 1997, pp. 106–111, also available from http://dynamo.ecn.purdue.edu/~givan.



Suresh Kalyanasundaram received his Bachelors degree in Electrical and Electronics Engineering and Masters degree in Physics from Birla Institute of Technology and Science, Pilani, India in 1996 and his Ph.D. degree from Purdue University in April 2000. Since then he has been working in the QoS research group at Motorola Labs. His research interests are in the area of performance analysis of wired and wireless networks.

Edwin K.P. Chong joined the School of Electrical and Computer Engineering at Purdue University in 1991, where he is currently an Associate Professor. He received the B.E.(Hons.) degree with First Class Honors from the University of Adelaide, South Australia, in 1987; and the M.A. and Ph.D. degrees in 1989 and 1991, respectively, both from Princeton University, where he held an IBM Fellowship. He received the NSF CAREER Award in 1995, and the ASEE Frederick Emmons Terman Award in 1998. He is a Senior Member of IEEE, and is chairman of the IEEE Control Systems Society Technical Committee on Discrete Event Systems. He has served on the editorial board of the IEEE Transactions on Automatic Control, and on various conference committees. His current interests are in communication networks and optimization methods. He coauthored a bestselling book, An Introduction to Optimization, Wiley-Interscience, 1996. He was named a Purdue University Faculty Scholar in 1999.



Ness B. Shroff received the B.S. degree from the University of Southern California, the M.S.E. degree from the University of Pennsylvania, and the M.Phil and Ph.D. degrees from Columbia University. He is currently an Associate Professor in the School of Electrical and Computer Engineering at Purdue University. During his doctoral study Dr. Shroff worked at AT&T Bell Labs (1991) and Bell Communications Research (1992), on problems involving fault management in telephone networks. His current research interests are in High Speed Broadband and Wireless Communication networks. He is especially interested in studying issues related to Performance Modeling and Analysis, Routing, Network Management, Scheduling, and Control in such networks. He also works on problems related to source coding, vector quantization, and error concealment. Dr. Shroff has received research and equipment grants to conduct fundamental work in broadband and wireless networks, and quantization from the National Science Foundation, AT&T, Hewlett Packard, Intel, LG Electronics, and the Purdue Research Foundation. He received the NSF CAREER award from the National Science Foundation in 1996. Dr. Shroff has served on the technical program committees of various conferences and on NSF review panels. He was the Conference Chair for the 14th Annual IEEE Computer Communications Workshop (CCW) and is Program Co-Chair for the High-Speed Networking Symposium at Globecomm 2000. He is also editor of IEEE Communication Letters and the Computer Networks journal.