ELSEVIER

Computer Networks 38 (2002) 613–630

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

Optimal resource allocation in multi-class networks with user-specified utility functions ☆

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

[a] *Global Telecom Solutions Sector, Motorola, Arlington Heights, IL 60004, USA*
[b] *Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523, USA*
[c] *School of Electrical and Computer Engineering, Purdue University, 1285 Electrical Engineering Building, West Lafayette, IN 47907, USA*

Received 18 April 2001; accepted 17 September 2001

Responsible Editor: E. Knightly

Abstract

In this work, we consider the problem of resource allocation in multi-class networks, where users specify the value they attach to obtaining different amounts of resource by means of a utility function. We develop a resource allocation scheme that maximizes the average aggregate utility per unit time. We formulate this resource allocation problem as a Markov decision process. We present numerical results that illustrate that our scheme performs better than the greedy resource allocation policy. We also discuss the implications of deliberate lying by users about their utility functions and develop a pricing scheme that prevents such lying. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Resource allocation; Pricing; Markov decision processes; Utility functions; Greedy scheme; Complete partitioning

1. Introduction

This work is motivated by two emerging trends in the networking area. One is the move towards multi-service networks, where a single network infrastructure is expected to support a variety of applications, such as data, voice, and video [1]. Secondly, applications are increasingly able to adapt to a wide range of resource amounts allocated to it by the network [2]. For example, phones can be equipped with multi-rate vocoders that can guarantee intelligible reception at the rates at which these vocoders operate. Scalable video compression schemes ensure successful playback at the receiver even at reduced bandwidth allocations [3]. Our aim in this work is to look at the resource allocation problem in this above scenario of multi-service networks and elastic applications.

For the successful operation of a multi-service network, several resources need to be allocated appropriately. The resources that are of interest in this context are bandwidth, buffer space, delay,

☆ This research was supported in part by the National Science Foundation through grants NCR-9624525, CDA-9617388, ECS-9501652, and ANI-9805441, and by the US Army Research Office through grant DAAH04-95-1-0246.

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

E-mail addresses: suresh.kalyanasundaram@motorola.com (S. Kalyanasundaram), echong@engr.colostate.edu (E.K.P. Chong), shroff@dynamo.ecn.purdue.edu, shroff@ecn.purdue.edu (N.B. Shroff).

PII: S1389-1286(01)00275-4

etc., which may or may not be allocated independently. In this work, we are mainly concerned with the *allocation of bandwidth as a resource*. While many of the ideas can be carried over to other types of resources as well, this is outside the scope of what we wish to address in this work. Therefore, we use the terms "resource" and "bandwidth" interchangeably in this work unless specified otherwise. In this work, we first present a solution to the problem of resource allocation on a single link and then show how it can be extended to the case of multiple links.

We now elaborate on the two emerging trends described above and discuss the implications they have for the resource allocation problem. With the need for multimedia services to be supported over the same network infrastructure, the resource allocator should consider the relative importance of a certain quantity of resource for various applications. In other words, the resource allocator should determine the application that would derive the maximum benefit from a certain quantity of resource, and then make a decision based on this information. For example, in general, delay-sensitive applications such as voice and video will benefit more from a certain quantity of bandwidth allocation than delay-tolerant file-transfer applications. The idea of *utility functions* from microeconomics captures this notion of importance of the quantity of resource allocated to the application very well [4]. We use a similar concept in our work, and the users specify the utility or value they attach to different quantities of resource using a utility function. We assume that the resource allocator knows the utility function of users at the time of resource allocation. The most likely way of achieving this is to have users communicate their utility functions to the resource allocator at the time of call setup. The resource allocator then allocates resources based on the objective of maximizing the aggregate average utility obtained per unit time. Maximization of aggregate utility is recognized as an important objective in resource allocation problems [5]. The objective of maximizing aggregate utility (as opposed to maximizing revenue) would be meaningful in private networks that are owned and operated by organizations for their own use. Even in the case of public networks, the potential loss of "goodwill" with customers might motivate operators to use the objective of maximizing aggregate utility. While other objectives may also be appropriate, we believe that maximizing aggregate utility is an important one to consider. Some of the ideas in this work would also readily extend to other objectives.

The second emerging trend of elastic applications means that the resource allocator has significant freedom in deciding the amount of resource that can be allocated to an application or connection. If the applications were non-elastic, as in the case of traditional voice telephony, the network can only decide whether or not the connection can be admitted. This is known as call admission control. But with elastic applications, besides doing call admission control, the network can also decide the amount of resource to allocate to an admitted call. The notion of utility functions becomes much richer and more useful for elastic applications. As we will see, the utility function of a non-elastic application reduces to the case of a "step function" because the application can gainfully use exactly a certain quantity of resource.

While applications are elastic, this does not necessarily mean that the resource allocated to a given connection or call can be altered during the course of the connection. We believe reallocating a different amount of resource during the lifetime of a connection will result in excessive overhead. We also believe that users will not like a frequent change in the quality of reception, which could result from frequent resource reallocations. Therefore, in this paper, we require that the resource is allocated for the entire lifetime of a connection and that resource reallocations are not allowed. More research is needed to determine if reallocations can be done during a connection's lifetime and how users perceive such reallocations. If reallocations can be done, the frequency of reallocations that users are willing to tolerate also needs to be determined. This issue will not be addressed in this paper.

Our approach to resource allocation in the above context is to formulate the problem as a Markov decision process (MDP). MDPs are used in dynamic probabilistic systems to make sequential decisions that optimize an appropriate objec-

tive [6–8]. In MDPs, the system can be in a finite number of states, and the decision maker can choose from several actions in each of those states. Once an action has been chosen in each state, the system evolves as a continuous-time Markov chain [1] and accumulates reward with time. The rate of reward accrual depends on the state of the system and the choice of control action in that state. In other words, associated with each state (and possibly depending on the control action chosen in that state), there is a reward accrual rate. This reward accumulates additively with time. In MDPs, the aim is to determine a policy (or equivalently, an action for each state) that optimizes the chosen objective. The chosen objective could either be a discounted total reward or an average per-unit-time reward. In the discounted total reward criterion, future rewards are discounted by a certain discount factor, which captures the notion of future rewards being less important than those accrued at the present time. In average reward criterion problems, the objective is to determine a policy that maximizes the average per-unit-time reward. (In our case, we wish to *maximize* the chosen objective because we associate "rewards" with each state. Alternatively, some authors associate "costs" with each state and *minimize* the chosen objective [9].)

While there has been a lot of work in the area of static resource allocation [5,10–14], where the number of users in the system is assumed fixed, our work is different in that we consider the dynamic case, where the state of the system changes with user arrivals and departures. For example, in Ref. [13], the authors discuss Pareto optimality and Nash equilibria for the case of a fixed number of users with known utility functions. Similarly, in Ref. [14] the authors consider the static scenario of a fixed number of elastic applications characterized by their minimum and peak rate requirements. In this framework, they obtain distributed implementations that converge to the optimal Nash bargaining solution. In our work, we exploit the knowledge of the arrival and departure pattern to make "informed" resource allocation decisions. We also implicitly take into consideration "opportunity costs" associated with decisions. The papers on static resource allocation cited above assume that whenever a call arrival or call departure occurs, resource is reallocated according to their static resource allocation algorithm again. As we elaborate later, our model does not allow resource reallocations to already existing users. Therefore, our scheme considers the future effects of its current decisions while allocating resources. MDPs have been successfully applied for admission control problems [15–18], but their application to resource allocation problems is new. (See the recent survey article [19].)

Recently, there has been work by Paschalidis and Tsitsiklis on a dynamic pricing framework for revenue and welfare maximization [20]. In this work the authors determine the optimal prices that maximize the long-run average revenue or welfare of the network. However, unlike our work which explicitly considers the case of elastic applications with the same user capable of operating over a range of bandwidth allocations, their work does not consider this scenario. Moreover, in Ref. [20], control of the users' access to the network is achieved indirectly through the pricing mechanism with the network admitting all users whose utility from the call admission is higher than the price as long as there is enough unused capacity. In contrast, we allow the network to have explicit control of its resources, and the network can determine whether or not to admit a call and the exact amount of resource to be allocated to the newly arriving call. While we also consider pricing in our work in Section 5, we use our pricing framework to ensure that users reveal their true utility functions rather than as a mechanism to control users' resource allocation amounts.

The rest of the paper is structured as follows. In Section 2, we formulate the single-link resource allocation problem as a continuous-time MDP. In Section 3, we discuss a few issues related to our resource allocation solution, such as incorporation of constraints on the call blocking probability, offline computation, complexity reduction, and extension of our scheme to the multiple-links case. In Section 4, we present sample numerical results

[1] Our description here is for the case of continuous-time MDPs. A similar description may also be given for the case of discrete-time MDPs.

and compare the performance of our optimal resource allocation scheme with that of a greedy scheme that takes a shortsighted view to the resource allocation problem. In Section 5, we discuss the implications of deliberate lying by users about their utility functions, and develop a pricing mechanism that prevents such behavior by users.

2. MDP formulation of resource allocation

In this section, we formulate the resource allocation problem as an MDP. Efficient methods exist to solve such problems, namely value iteration, policy iteration, and linear programming [6–8]. In the interest of space, we merely formulate the problem as an MDP and refer the reader to the references cited above for solution techniques.

We first describe the problem scenario. In this section, we consider a single link with a finite capacity that supports a variety of applications and consider the allocation of bandwidth to calls or connections. We extend our resource allocation scheme to the case of multiple links in Section 3.6. This problem of bandwidth allocation is relevant to both circuit-switched and packet-switched networks. In packet-switched networks, the notion of bandwidth allocation hides the packet-level dynamics and simplifies analysis [21]. We assume that allocations can only be made in multiples of a certain minimum quantity, say, Δ bandwidth units (BWUs). This assumption is not very restrictive from a practical viewpoint because it is true for many slotted systems. Besides, we can make Δ as small as we wish. Also, without loss of generality, we assume that $\Delta = 1$ BWU. We make this assumption to reduce notational complexity. We let the capacity of the channel be C BWUs. Connections or calls can be allocated any integer amount of resource between zero and C BWUs.

Utility functions give the amount of utility or benefit or gain that the user or application obtains for different amounts of allocated resource. It is a measure of how "happy" or "satisfied" a user or application is at receiving a certain amount of allocation. Economists use 'utils' as units of the utility values. Typically, if resources are infinitely divisible, the utility function would be a mapping from the interval $[0, C]$ to the non-negative reals $\mathbb{R}^+$. But, in our case, it suffices if the utility function specifies the per-unit-time utility obtained for all feasible allocations. Note that the only feasible allocations are the integers between zero and C. We now introduce the concept of utility functions in this limited context where only a finite number of allocations are possible. The domain of the utility function is the set of all integers between zero and C. We let $U(i)$ for $i = 0, 1, 2, \ldots, C$ denote the value of the utility function for an allocation of i BWUs. The value of $U(i)$ gives the utility obtained by the application for an allocation of i BWUs for one second and has units of utils/s. Thus, a utility function $\boldsymbol{U}$ for the purpose of resource allocation is a vector $[U(0), U(1), \ldots, U(C)]$. We further assume that the per-unit-time utility (or utils per second) that the application or user receives has to be a multiple of a certain other minimum quantity, say, δ utils/s. Again, without loss of generality, we assume that $\delta = 1$ to avoid needless notational complexity. We also require that the utility values stay uniformly bounded. In other words, there exists a $U_{\max}$ such that $U(i) \leqslant U_{\max}$ for $i = 0, 1, \ldots, C$ for all users. Utility functions also satisfy the condition

$$U(i) \geqslant 0 \quad \text{for } i = 0, 1, 2, \ldots, C.$$

The above condition states that applications or users cannot obtain a negative utility from a positive quantity of allocated resource. Because utility functions are non-negative, bounded, and integer valued, we have

$$U(i) \in \{0, 1, 2, \ldots, U_{\max}\} \quad \text{for } i = 0, 1, 2, \ldots, C, \tag{1}$$

where we assume $U_{\max}$ is an integer. It is also natural to assume that

$$U(0) = 0, \tag{2}$$

where Eq. (2) simply states that users cannot have a non-zero utility for zero resource allocation.

Besides these properties, there are two others that many utility functions satisfy. However, our problem formulation does not depend on whether or not the following conditions are met. First, the utility function is expected to be non-decreasing, i.e.,

$$U(i) \leqslant U(i+1) \quad \text{for } i = 0, 1, \ldots, C-1.$$

This is due to the fact that applications cannot be expected to derive more benefit from a smaller amount of resource allocation than from a larger amount of allocation. Secondly, many example utility functions satisfy the following concavity property, which is a consequence of the "law of diminishing returns":

$$U(j) - U(j-1) \geqslant U(i) - U(i-1) \quad \text{for all } j < i.$$

The above equation says that for higher resource allocations the utility increments obtained are lower. Applications, typically, find the most use for the initial allocations, and as the allocation is increased, the benefits begin to diminish. Some utility functions do not have the concavity property. (See Ref. [2] and the example of the multi-rate vocoder below.) Therefore, our scheme is designed to work with non-concave utility functions as well.

A very elastic application can gainfully use every additional amount of allocated resource, i.e., $U(i) > U(i-1)$ for $i = 1, 2, \ldots, C$. File-transfer applications, which are very elastic, can be expected to have such utility functions. If a phone is equipped with a multi-rate vocoder that can operate at rates r_1, r_2, and r_3 with $r_1 < r_2 < r_3 \leqslant C$, then its utility function will be such that $U(r_3) > U(r_2) > U(r_1)$, and $U(i)$ for all other values of allocated resource will be given as follows:

$$U(i) = \begin{cases} 0 & \text{for } 0 \leqslant i < r_1, \\ U(r_1) & \text{for } r_1 \leqslant i < r_2, \\ U(r_2) & \text{for } r_2 \leqslant i < r_3, \\ U(r_3) & \text{for } r_3 \leqslant i \leqslant C. \end{cases} \tag{3}$$

A non-elastic application that can only operate at a rate of r BWUs has a step utility function of the following form:

$$U(i) = \begin{cases} 0 & \text{for } i < r, \\ U(r) & \text{for } r \leqslant i \leqslant C. \end{cases}$$

We let $\mathscr{U}$ denote the set of all possible utility functions $\boldsymbol{U}$. Note that the set of all allowed utility functions is limited by the conditions that a utility function has to satisfy, i.e.,

$$\mathscr{U} = \{\boldsymbol{U} = [U(0), U(1), \ldots, U(C)]: \boldsymbol{U} \text{ satisfies Eqs. (1) and (2)}\}.$$

The set $\mathscr{U}$ has only a finite number of utility functions in it. It can be verified that the number of elements in $\mathscr{U}$ is no greater than $(U_{\max}+1)^C$, where $U_{\max}$ is the greatest utility that a user can obtain for any resource allocation amount. We let M denote the total number of elements in the set $\mathscr{U}$. We think of all users having the same utility function $U \in \mathscr{U}$ as belonging to a class. We also uniquely number the utility functions in $\mathscr{U}$ from one through M, and use the notation $\boldsymbol{U}_k = [U_k(0), U_k(1), \ldots, U_k(C)]$ to denote the kth such utility function. Note that once we specify the index of the utility function k, the quantities $U_k(i)$ get fixed for all $i = 0, 1, 2, \ldots, C$.

We assume that class-k users arrive according to an independent Poisson process with rate λ_k, and that the call holding duration of a class-k call allocated i BWUs is independent and identically distributed (i.i.d.) exponential with mean $1/\mu_{ki}$. By making the call holding duration depend on the allocation, we account for the scenario where users have a fixed amount of data to send resulting in users having a longer call holding duration for smaller allocations than for larger allocations. Given the above scenario, the operation of the network is as follows. When a new user arrives, say, belonging to class k, the network allocates it i BWUs of resource, where $0 \leqslant i \leqslant C$. (When $i = 0$, the network blocks the newly arriving call.) The network then starts accruing additional utility (or reward) at the rate of $U_k(i)$ utils/s until the call gets completed. The resource allocation problem is to determine i, given the state of the system, $\boldsymbol{U}_k$, λ_k, and μ_{ki} for $k = 1, 2, \ldots, M$ and $i = 1, 2, \ldots, C$, that maximizes the average aggregate utility per unit time of the system. By formulating the problem as an MDP, we determine what appropriate resource allocation decisions should be made to achieve such an objective.

We denote the state of the system by a matrix $\boldsymbol{N}$ with M rows and C columns. The term on the kth row and the ith column of the matrix $\boldsymbol{N}$ is denoted by n_{ki}, which is the number of class-k users allocated i BWUs. Let

$$n_k = \sum_{i=1}^{C} n_{ki} \quad \text{for } k = 1, 2, \ldots, M \tag{4}$$

denote the number of active class-k users in the system. The finite capacity (C) of the link implies that any valid state of the system has to satisfy the condition

$$\sum_{k=1}^{M}\sum_{i=1}^{C} in_{ki} \leqslant C.$$

The state space S is the set of all possible states in the system, which can be written as follows:

$$S = \left\{ \boldsymbol{N} = [n_{ki}]\colon 1 \leqslant k \leqslant M, 1 \leqslant i \leqslant C, n_{ki} \in \mathbb{Z}^{+} \text{ and } \sum_{k=1}^{M}\sum_{i=1}^{C} in_{ki} \leqslant C \right\},$$

where $\mathbb{Z}^{+}$ is the set of all non-negative integers. Any n_{ki} for $1 \leqslant k \leqslant M$ and $1 \leqslant i \leqslant C$ can only take integer values between zero and $\lfloor C/i \rfloor$, where $\lfloor x \rfloor$ is the greatest integer not exceeding x. Therefore, the number of values that each n_{ki} can take is finite. This implies that the state space S itself has a finite number of elements in it.

In each state of the system, the resource allocator has a choice of several actions. An action specifies the amount of resource to be allocated to a new call arrival of each call class. If a call arrival occurs, the resource allocator allocates the amount of resources specified by the chosen action in that state, and the system moves to the corresponding next state. If a call departure occurs in that state, then the system moves to the corresponding next state. In either case, the whole process gets repeated again in the new system state. The objective in our resource allocation problem is to determine the optimal action for each state that maximizes the average utility per unit time.

We denote an action in state $\boldsymbol{N} \in S$ by an M-vector $\boldsymbol{u} = [u_1, u_2, \ldots, u_M]$ such that $u_k \in \{0, 1, 2, \ldots, C\}$. The value of u_k denotes the number of units of resource that a newly arriving class-k user is to be allocated. (When $u_k = 0$, the resource allocator rejects newly arriving class-k users, and the state of the system does not change.) Clearly, the number of allowed actions in a state depends on the state of the system. For example, in a state that already has a non-zero amount of resource in use, a newly arriving user cannot be allocated C BWUs of resource. The set of all allowed actions in state $\boldsymbol{N}$, known as the action space, can be written as follows:

$$K_{\boldsymbol{N}} = \left\{ \boldsymbol{u}\colon u_k \in \{0, 1, \ldots, C\} \text{ and } \sum_{k=1}^{M}\sum_{i=1}^{C} in_{ki} + \max\{u_1, u_2, \ldots, u_M\} \leqslant C \right\},$$

where the n_{ki} for $1 \leqslant k \leqslant M$ and $1 \leqslant i \leqslant C$ define the matrix $\boldsymbol{N}$. The action space in a state is the set of all possible actions that does not exceed the capacity of the channel, and it follows that the action space $K_{\boldsymbol{N}}$ for $\boldsymbol{N} \in S$ has only a finite number of elements in it. If a new class-k call arrives in state $\boldsymbol{N} \in S$ and $\boldsymbol{u}$ is the chosen action, then it is allocated u_k units of resource. The system then moves to the appropriate next state. If there is a call completion, the amount of resource used by the terminating call gets released for future call arrivals.

In our MDP formulation of the resource allocation problem, we now specify the reward rate in each state $\boldsymbol{N} \in S$ when action $\boldsymbol{u} \in K_{\boldsymbol{N}}$ is chosen. The reward rate is

$$R_{\boldsymbol{N}}^{\boldsymbol{u}} = \sum_{k=1}^{M}\sum_{i=1}^{C} n_{ki} U_k(i) \text{ utils/s}, \tag{5}$$

which is the sum of the utility values of each active user in the system corresponding to the amount of resources allocated to it. Note that the definition of reward in Eq. (5) is independent of the choice of action $\boldsymbol{u}$ in that state.

To complete our MDP formulation of the resource allocation problem, we now specify the transition rates of the system. We denote the transition rate from state $\boldsymbol{N} \in S$ to state $\boldsymbol{Q} \in S$ when action $\boldsymbol{u} \in K_{\boldsymbol{N}}$ is chosen by $\alpha_{\boldsymbol{N},\boldsymbol{Q}}^{\boldsymbol{u}}$. Let $\boldsymbol{N} = [n_{ki}]$ and $\boldsymbol{Q} = [q_{ki}]$. For ease of description, we also define $\boldsymbol{\delta}_{ki}$ to be an $M \times C$ matrix with zeros in all but the (k,i)th entry where it has a one. The transition rates are then given as follows:

$$\alpha_{\boldsymbol{N},\boldsymbol{Q}}^{\boldsymbol{u}} = \begin{cases} \lambda_k & \text{if } u_k \neq 0 \text{ and } \boldsymbol{Q} = \boldsymbol{N} + \boldsymbol{\delta}_{ku_k}, \\ n_{ki}\mu_{ki} & \text{if } n_{ki} \neq 0 \text{ and } \boldsymbol{Q} = \boldsymbol{N} - \boldsymbol{\delta}_{ki}, \\ 0 & \text{otherwise.} \end{cases}$$

Given that the action in state $\boldsymbol{N}$ admits class-k calls, i.e., $u_k \neq 0$, the system moves to a state with one additional class-k call at the rate of λ_k, which is the call arrival rate of class-k calls. If a new class-k call arrives, then the new call arrival is allocated u_k BWUs, and hence the number of active class-k users allocated u_k BWUs increases by one. With n_{ki} active class-k users allocated i BWUs, the system moves to the state with one less class-k call allocated i units of resource at the rate of $n_{ki}\mu_{ki}$. In other words, given that action $\boldsymbol{u}$ is chosen in state $\boldsymbol{N}$, the system stays in the same state for an exponentially distributed duration of time with mean $[\sum_{k=1}^{M}(\lambda_k I_{\{u_k\neq 0\}} + \sum_{i=1}^{C} n_{ki}\mu_{ki})]^{-1}$, where n_k is as defined in Eq. (4), and I is the indicator function given by

$$I_{\{u_k\neq 0\}} = \begin{cases} 1 & \text{if } u_k \neq 0, \\ 0 & \text{if } u_k = 0. \end{cases}$$

When the system makes a transition, it moves to state $\boldsymbol{N} + \boldsymbol{\delta}_{ku_k}$ with probability

$$\frac{\lambda_k I_{\{u_k\neq 0\}}}{\sum_{j=1}^{M}\left(\lambda_j I_{\{u_j\neq 0\}} + \sum_{i=1}^{C} n_{ji}\mu_{ji}\right)},$$

and to state $\boldsymbol{N} - \boldsymbol{\delta}_{ki}$ with probability

$$\frac{n_{ki}\mu_{ki}}{\sum_{j=1}^{M}\left(\lambda_j I_{\{u_j\neq 0\}} + \sum_{i=1}^{C} n_{ji}\mu_{ji}\right)}.$$

It follows from the above formulation that we have a continuous-time MDP to solve because the transition rates and the reward rates depend only on the current state and the chosen action in that state. Note that an MDP is completely characterized by its state and action spaces, reward rates, and transition rates. This continuous-time MDP formulation with average reward optimization criterion has finite state space and finite action space. Efficient methods exist to solve such problems, namely value iteration, policy iteration, and linear programming [6–8]. For the sake of completeness, we briefly describe one particular solution technique, namely linear programming, that was also used in Section 4 to obtain the numerical results. To obtain the optimal policy we solve the following linear program:

$$\begin{aligned} \underset{\{x_N^{\boldsymbol{u}}:\boldsymbol{N}\in S \text{ and } \boldsymbol{u}\in K_N\}}{\text{maximize}} \quad & \sum_{\boldsymbol{N}\in S}\sum_{\boldsymbol{u}\in K_N} R_N^{\boldsymbol{u}} x_N^{\boldsymbol{u}}, \\ \text{subject to} \quad & \sum_{\boldsymbol{Q}\in S}\sum_{\boldsymbol{u}\in K_N} x_N^{\boldsymbol{u}}\alpha_{N,\boldsymbol{Q}}^{\boldsymbol{u}} \\ & \quad - \sum_{\boldsymbol{Q}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{Q}}} \alpha_{\boldsymbol{Q},N}^{\boldsymbol{u}} x_{\boldsymbol{Q}}^{\boldsymbol{u}} = 0 \quad \text{for } \boldsymbol{N}\in S, \\ & \sum_{\boldsymbol{N}\in S}\sum_{\boldsymbol{u}\in K_N} x_N^{\boldsymbol{u}} = 1, \quad x_N^{\boldsymbol{u}} \geqslant 0 \quad \text{for } \boldsymbol{N}\in S. \end{aligned} \tag{6}$$

Once the optimal $x_N^{\boldsymbol{u}*}$ have been obtained, the optimal action in each state is obtained according to the following procedure. Let S^* be the set

$$S^* = \left\{\boldsymbol{N}\in S : x_N^{\boldsymbol{u}*} > 0 \text{ for some } \boldsymbol{u}\in K_N\right\}. \tag{7}$$

Then the optimal action in state $\boldsymbol{N}\in S^*$ is determined as follows. For $\boldsymbol{N}\in S^*$, action v is chosen with probability $x_N^{v*}/\sum_{\boldsymbol{u}\in K_N} x_N^{\boldsymbol{u}*}$, and for $\boldsymbol{N}\notin S^*$, an arbitrary action $v\in K_N$ is chosen with probability one. Note that it follows from Ref. [22] that there exists an optimal policy such that $x_N^{\boldsymbol{u}*} > 0$ for exactly one action $\boldsymbol{u}\in K_N$ for each state $\boldsymbol{N}\in S^*$.

An important special case of our formulation above is the admission control problem for networks with non-elastic applications. Such problems have been the subject of several papers [15–18]. When applications are non-elastic, our resource allocation problem reduces to the admission control problem, where the network either rejects the call or admits the call and allocates it the exact amount of resource it needs. The solution developed here can be used for the admission control problem as well.

3. Discussion

In this section, we discuss a few aspects related to the MDP formulation of the resource allocation problem developed in the last section.

3.1. Offline computation

The computation of the optimal action associated with each state (or the optimal policy) can be done beforehand and need not be done each time a call arrival or call departure occurs. The computed decisions may be stored in the form of a table, and

a table look-up can be done each time the system reaches a new state. Thus there is little real-time computational overhead involved in the implementation of the scheme. However, if the call arrival rates and the call holding durations vary with time, the optimal decisions may need to be recomputed as needed based on updated values of these quantities. (Numerical results in Section 4 show that the optimal decision depends on the value of the call arrival rates and call holding durations of call classes.)

3.2. Uncertainty in transition rates

As we show in Section 4, the optimal policy depends on the values of the call arrival rates and call holding durations of call classes. But the resource allocator may not have exact knowledge of these quantities beforehand. However, it is reasonable to expect the resource allocator to know a range over which these quantities take values. Given such a range of possible values for the call arrival rates and the call holding durations, the methods described in Refs. [23–25] 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 utility 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. In other words, using the max–min optimal policy, the resource allocator can maximize the average utility per unit time assuming that the transition rates will get chosen in a manner that minimizes this quantity.

3.3. Model reduction techniques

While our scheme can be applied to network scenarios with 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. In this section, we describe a few model reduction techniques that can be used to reduce the computational complexity involved in obtaining the optimal solution. We classify the model reduction techniques in this section into those that yield the optimal policy and those that can be used to obtain approximately optimal policies.

We first describe a model reduction technique such that the solution to the reduced model still yields the optimal policy. We make a simple observation that can result in a significant reduction in the state and action spaces of practical problems of interest. Note that if a certain utility function $\boldsymbol{U}_k$ is such that $U_k(i+1) = U_k(i)$, then the network obtains no additional amount of utility by allocating that user $i+1$ BWUs instead of i BWUs of resource. Therefore, we can eliminate those states and actions that correspond to allocating class-k users $i+1$ BWUs. For example, the only allocations that are meaningful for the multi-rate vocoder whose utility function is given by Eq. (3) are r_1, r_2, and r_3. We can define the set of all actions that can be eliminated in state $\boldsymbol{N} \in S$ due to the above observation as follows:

$$L_{\boldsymbol{N}} = \{\boldsymbol{u} \in K_{\boldsymbol{N}}: u_k > 0 \text{ and } U_k(u_k) = U_k(u_k - 1)\}.$$

Similarly, we can define the set of states that can be eliminated due to the above observation as follows:

$$A = \{\boldsymbol{N} \in S: n_{ki} > 0 \text{ and } U_k(i) = U_k(i-1) \text{ for } i \geqslant 1\}.$$

Based on the above observation, we can now define our reduced state space as

$$S' = S \setminus A,$$

where the notation $S \setminus A$ is used to denote the set of all elements in S but not in A. Similarly, the reduced action space in state $\boldsymbol{N} \in S'$ can be written as

$$K'_{\boldsymbol{N}} = K_{\boldsymbol{N}} \setminus L_{\boldsymbol{N}}.$$

We believe that the above state and action space reduction will reduce the computational complexity involved in obtaining the solution to practical problems of interest.

We now describe model reduction techniques that yield approximately optimal policies. Because,

we compromise on the optimality of the policies we seek, the reduction in complexity can be quite substantial. Generic model reduction techniques [26,27] can be used to obtain approximate solutions that reduce the computational complexity involved in solving MDPs with large state and action spaces. Neuro-dynamic programming [9] techniques have recently been successfully applied to admission control problems to obtain approximate solutions [28].

We next describe the *optimal complete partitioning policy*, which is an approximately optimal policy. The optimal complete partitioning policy would set aside portions of the bandwidth exclusively for users of different classes (similar to the complete partitioning admission control policy [29]). Let there be partitions $P_1, P_2, \ldots, P_M$ of the capacity of the channel C such that

$$\sum_{k=1}^{M} P_k = C,$$

with $P_k \geqslant 0$ being the capacity of the channel that is exclusively reserved for use by class-k calls. Because we allow only integer allocations to users, P_k for $k = 1, 2, \ldots, M$ have to be integers. The aim of the optimal complete partitioning policy is to determine the values $P_1^*, P_2^*, \ldots, P_M^*$ such that the average aggregate utility is maximized. Because the set of complete partitioning policies are a subset of the set of all policies we considered in Section 2, the average aggregate utility obtained from the optimal complete partitioning policy can at most be equal to that of the optimal policy obtained in Section 2.

Within each partition P_k reserved for class-k calls the optimal choice of allocations can be determined by solving MDPs that have much smaller state and action spaces. Let $R_k^*(P_k)$ for $k = 1, 2, \ldots, M$ be the optimal average aggregate utility from class-k calls when the capacity set aside for class-k calls is P_k. The value for $R_k^*(P_k)$ is obtained from the solution to the MDP that is formulated in the same manner as in Section 2 but with a single call class and with the capacity being equal to P_k. The optimal complete partitioning policy can then be cast in the following form:

$$\begin{aligned} \underset{P_1,P_2,\ldots,P_M}{\text{maximize}} \quad & \sum_{k=1}^{M} R_k^*(P_k), \\ \text{subject to} \quad & \sum_{k=1}^{M} P_k = C, \\ & P_k \geqslant 0, \\ & P_k \text{ integer}. \end{aligned}$$

The solution to the above problem can be obtained by a dynamic program similar to that in Ref. [29]. If we let $f_k(y)$ be the average aggregate utility of classes 1 through k with y as the capacity available, then the dynamic programming equations are

$$\begin{aligned} & f_1(y) = R_1^*(y) \quad \text{for } 0 \leqslant y \leqslant C, \\ & f_k(y) = \underset{0 \leqslant s \leqslant y}{\text{maximize}} \ (R_k^*(s) + f_{k-1}(y - s)) \\ & \qquad \text{for } 0 \leqslant y \leqslant C \text{ and } k = 2, 3, \ldots, M. \end{aligned}$$

The solution to the above dynamic program would give the optimal partition $[P_1^*, P_2^*, \ldots, P_M^*]$.

The performance of the optimal complete partitioning policy should be compared with that of the optimal policy over a range of parameter values to determine the loss in utility that can be expected from using the optimal complete partitioning policy. Another strategy would be to use the policy that performs best between the optimal complete partitioning policy and the greedy policy that is explained in Section 4. In Section 4, we compare the performance of the optimal complete partitioning policy with that of the optimal policy. In our example, we find that the performance of the optimal complete partitioning policy is fairly close to that of the optimal policy. While we have discussed some complexity reduction techniques in this section, more work is needed to study the effectiveness of these in practical problems of interest and to develop techniques to reduce the complexity even further.

3.4. Maximization of utility with constraints

Our resource allocation scheme was developed with the objective of maximizing utility. In certain situations, maximization of utility will not be the only objective of the network. For example, in

admission control problems with non-elastic applications, several authors consider constraints placed on the call blocking probabilities experienced by the call classes [15,17,18]. In Ref. [15], the authors consider the scenario where each call class has a maximum call blocking probability that it can tolerate. Therefore, they design an admission control algorithm that maximizes the throughput of the network subject to the constraints placed on the call blocking probabilities of each call class. The viewpoint in Refs. [17,18] is that different levels of pricing will require that the call blocking probabilities of call classes be related in a certain manner to justify the difference in prices. The authors treat the problem of maximizing throughput and maintaining the desired relations between call blocking probabilities as a multi-objective optimization problem.

Motivated by the same considerations as in the above papers, we envision a scenario where the resource allocation needs to consider multiple objectives. Constraints similar to those in Refs. [15,17,18] can be incorporated in our MDP formulation. For example, if each call class has a maximum call blocking probability of D_k, $1 \leqslant k \leqslant M$ that it can tolerate, then the optimal decision is obtained from the solution of the following modified linear program:

$$\begin{aligned} \underset{\{x_{\boldsymbol{N}}^{\boldsymbol{u}}:\boldsymbol{N}\in S \text{ and } \boldsymbol{u}\in K_{\boldsymbol{N}}\}}{\text{maximize}} \quad & \sum_{\boldsymbol{N}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{N}}} R_{\boldsymbol{N}}^{\boldsymbol{u}} x_{\boldsymbol{N}}^{\boldsymbol{u}}, \\ \text{subject to} \quad & \sum_{\boldsymbol{Q}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{N}}} x_{\boldsymbol{N}}^{\boldsymbol{u}} \alpha_{\boldsymbol{N},\boldsymbol{Q}}^{\boldsymbol{u}} \\ & \quad - \sum_{\boldsymbol{Q}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{Q}}} \alpha_{\boldsymbol{Q},\boldsymbol{N}}^{\boldsymbol{u}} x_{\boldsymbol{Q}}^{\boldsymbol{u}} = 0 \quad \text{for } \boldsymbol{N}\in S, \\ & \sum_{\boldsymbol{N}\in S}\sum_{\boldsymbol{u}\in K_{\boldsymbol{N}}} x_{\boldsymbol{N}}^{\boldsymbol{u}} = 1, \; x_{\boldsymbol{N}}^{\boldsymbol{u}} \geqslant 0 \quad \text{for } \boldsymbol{N}\in S. \end{aligned} \tag{8}$$

$$\sum_{\boldsymbol{N}\in S}\sum_{\boldsymbol{u}\in B_{\boldsymbol{N}}(k)} x_{\boldsymbol{N}}^{\boldsymbol{u}} \leqslant D_k \quad \text{for } 1 \leqslant k \leqslant M, \tag{9}$$

where $B_{\boldsymbol{N}}(k)$ is the following set:

$$B_{\boldsymbol{N}}(k) = \{\boldsymbol{u} = [u_1, u_2, \ldots, u_M] : \boldsymbol{u} \in K_{\boldsymbol{N}} \text{ and } u_k = 0\}.$$

The set $B_{\boldsymbol{N}}(k)$ is the set of all actions in state $\boldsymbol{N}$ that reject class-k calls, and Eq. (9) is the long-run average duration of time spent in those states of the system that reject class-k calls. We note that $x_{\boldsymbol{N}}^{\boldsymbol{u}}$ represents the long-run fraction of time the system spends in state $\boldsymbol{N}$ with $\boldsymbol{u}$ as the chosen action. Once the optimal $x_{\boldsymbol{N}}^{\boldsymbol{u}*}$ have been obtained from the above linear program, the optimal action in each state is obtained according to the procedure described in Section 2. Using the result in Ref. [22] it can be shown that the optimal solution obtained in such a manner will require randomization in at most M states.

3.5. Extension to general call holding times

In our work, we assume 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 [30]. Our work here is a first step towards addressing the resource allocation problem with 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 [31,32]. Thus, if the optimal resource allocation policy obtained by our method turns out to be coordinate convex, then that policy will continue to be optimal among all coordinate convex policies for any general call holding distribution with the same mean as the original exponential call holding duration. Our policy will continue to be optimal among all coordinate convex policies for any general call holding distribution with the same mean as the original exponential call holding duration whenever the optimal policy itself is coordinate convex. However, we cannot guarantee the optimality of our resource allocation policy among all resource allocation 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 [33]. 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.

3.6. Multi-link resource allocation

In this section, we extend our single-link resource allocation solution to the case of multiple links with pre-computed routes. The objective of the resource allocation is to maximize the aggregate utility per unit time of the entire network. The utility functions now denote the amount of utility that the user obtains from all possible end-to-end resource allocations. If a user obtains different amounts of allocation over different links through which it traverses, then the utility that the user obtains corresponds to that obtained from the minimum of those allocations. We consider a network with a finite set $\mathscr{L}$ of links, and we assume that the routes for each user are pre-computed and fixed. Because the number of links in the network is finite, so are the number of possible (cycle-free) routes. For the multi-link case, two users are said to belong to the same class only if those users have identical utility functions (as defined in Section 2) and identical routes. Because we have a finite number of possible utility functions and finite number of routes, we have a finite number of possible classes. We denote the total number of classes of users by M'. We assume that the route information for user classes is stored in a matrix γ such that

$$\gamma_{kj} = \begin{cases} 1 & \text{if class-}k\text{ users are routed through link } j, \\ 0 & \text{if class-}k\text{ users are not routed through link } j. \end{cases} \tag{10}$$

We denote the capacity of link j by C_j. As in the single-link case, we assume that class-k users arrive according to a Poisson process with rate λ_k and that the call holding duration of a class-k call allocated i BWUs is independent and exponentially distributed with a mean of $1/\mu_{ki}$. For the multi-link case, states are again denoted by a matrix $\boldsymbol{N}$ with M' rows and $\max\{C_j: j \in \mathscr{L}\}$ columns. We let

$$C_{\max} = \max\{C_j: j \in \mathscr{L}\},$$

where we assume that $C_{\max}$ is an integer. The term on the kth row and the ith column of the matrix $\boldsymbol{N}$ is denoted by n_{ki}, which is the number of class-k users allocated i BWUs. Note that the user is allocated i BWUs over all the links through which the class-k user traverses. The state space S for the multi-link case can be written as

$$S = \left\{ \boldsymbol{N} = [n_{ki}] : n_{ki} \in \mathbb{Z}^+ \text{ and } \sum_{k=1}^{M'} \sum_{i=1}^{C_{\max}} i n_{ki} \gamma_{kj} \leqslant C_j \text{ for } j \in \mathscr{L} \right\}.$$

Note that any valid state has to be such that the link capacity of none of the links is exceeded. The γ_{kj} in the above equation ensures that the summation is carried out only over those classes of users that are routed through link j. Actions are denoted by an M' vector $\boldsymbol{u} = [u_1, u_2, \ldots, u_{M'}]$ such that $u_k \in \{0, 1, \ldots, C_{\max}\}$. As before, the value of u_k denotes the number of units of resource to be allocated to a class-k user. The action space can now be written as follows:

$$K_{\boldsymbol{N}} = \left\{ \boldsymbol{u} : \sum_{k=1}^{M'} \sum_{i=1}^{C_{\max}} i n_{ki} \gamma_{kj} + \max\{u_1\gamma_{1j}, u_2\gamma_{2j}, \ldots, u_M\gamma_{Mj}\} \leqslant C_j \text{ for all } j \in \mathscr{L} \right\}.$$

The reward rates and the transition rates are defined as in Section 2. The problem can then be solved by any of the methods used to solve MDPs [6–8].

4. Numerical results

In this section, we present numerical results for the single-link case to illustrate our resource allocation scheme. In our first example the capacity of the channel is 4 BWUs. There are two classes of calls with their utility functions given by $\boldsymbol{U}_1 = [1, 2, 2, 2]$ and $\boldsymbol{U}_2 = [0, 3, 3, 3]$. Calls belonging to class 1 are elastic and can use 1 or 2 BWUs gainfully with varying utilities associated with it. Class-2 calls, however, are non-elastic and can

only use 2 BWUs. Any amount of allocation above 2 BWUs does not result in an increase in utility, and any amount of allocation below 2 BWUs results in zero utility for class-2 calls. The call holding durations of both classes of calls are assumed i.i.d. exponential with mean 60 s.

The optimal action in each state of the system for the above example is given in Table 1. We consider three different scenarios by varying the call arrival rate of class-2 calls. We put an 'x' for the optimal action in those states that are "do not care" states. These states do not belong to S^* as defined in Eq. (7). Thus any arbitrary action can be chosen in such states. From the table, it can be seen that such states are transient states corresponding to the chosen policy. Those states that are not listed in the table are "do not care" states for all the three scenarios considered. The results in the table are intuitively appealing. For example, as we increase the call arrival rate of class-2 calls, the optimal resource allocation policy tends to favor class-2 calls over class-1 calls. This is to be expected because allocating 2 BWUs to a class-2 call results in utility being accrued at the rate of 1.5 utils/s per BWU, which is higher than that obtained for class-1 calls for any amount of allocation, and also due to the total capacity being an even number.

Table 1
Optimal actions at the corresponding states for three different scenarios

State	$\lambda_1 = 0.5$, $\lambda_2 = 0.01$	$\lambda_1 = 0.5$, $\lambda_2 = 0.05$	$\lambda_1 = 0.5$, $\lambda_2 = 0.1$
$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$	(2,2)	(2,2)	(0,2)
$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$	(2,2)	(0,2)	(0,2)
$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{pmatrix}$	(0,0)	(0,0)	(0,0)
$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$	(2,2)	(0,2)	x
$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$	(0,0)	(0,0)	x
$\begin{pmatrix} 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$	(0,0)	x	x

The utility function of class-1 calls is $[1, 2, 2, 2]$ and that of class-2 calls is $[0, 3, 3, 3]$.

Table 2
Optimal actions at the corresponding states for three different scenarios

State	$\lambda_1 = 0.1$, $\lambda_2 = 0.02$	$\lambda_1 = 0.1$, $\lambda_2 = 0.03$	$\lambda_1 = 0.1$, $\lambda_2 = 0.3$
$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$	(3,2)	(3,2)	(0,2)
$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$	(1,2)	(0,2)	(0,2)
$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{pmatrix}$	(0,0)	(0,0)	(0,0)
$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$	(3,0)	(3,2)	x
$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$	(1,0)	(1,0)	x
$\begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$	(0,2)	(0,2)	x
$\begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$	(0,0)	(0,0)	x
$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$	(1,0)	(1,0)	x
$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$	(0,0)	(0,0)	x

The utility function of class-1 calls is $[1, 1, 4, 4]$ and that of class-2 calls is $[0, 3, 3, 3]$.

To further illustrate our scheme, we consider a second example. As before, we have two call classes with the channel capacity being 4 BWUs. But now the utility function for class-1 calls is $[1, 1, 4, 4]$ and that of class-2 calls is $[0, 3, 3, 3]$. Call holding durations of both call classes are assumed i.i.d. exponential with a mean of 60 s. Table 2 gives the optimal actions for the corresponding states for three different scenarios.

In Fig. 1, we plot the average utility per unit time obtained from our utility maximizing scheme. The capacity of the channel is 20 BWUs, and there are two call classes with utility functions $[1, 1, 4, 4, \ldots, 4]$ and $[0, 3, 3, \ldots, 3]$, respectively. The figure is for a fixed λ_2 value of 0.05 calls/s, and the call holding durations of both classes of calls are assumed i.i.d. exponential with mean 60 s. We compare the performance of our scheme with the greedy scheme. The greedy scheme allocates all incoming calls that amount of resource of the

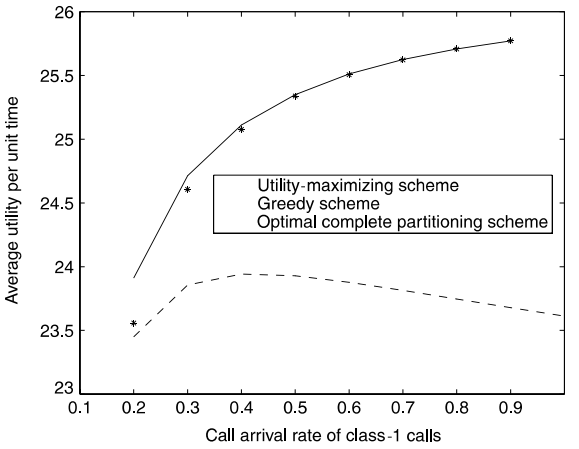

Fig. 1. Comparison of the average utility per unit time obtained for the greedy, utility maximizing, and optimal complete partitioning schemes. The utility function of class-1 calls is $[1, 1, 4, 4, \ldots, 4]$ and that of class-2 calls is $[0, 3, 3, \ldots, 3]$.

unused capacity that maximizes the utility obtained from the incoming call. To exactly characterize the action chosen by the greedy policy in state $\boldsymbol{N} \in S$, we define the set $\mathscr{F}_{\boldsymbol{N}}$ of all feasible allocations in state $\boldsymbol{N}$ as follows:

$$\mathscr{F}_{\boldsymbol{N}} = \left\{ x \in \mathbb{Z}^+ : x. + \sum_{k=1}^{M} \sum_{i=1}^{C} in_{ki} \leqslant C \right\}.$$

With the above definition, the kth component of the action chosen by the greedy scheme is

$$u_k(\boldsymbol{N}) = \min\{x \in \mathscr{F}_{\boldsymbol{N}} : U_k(x) \geqslant U_k(y) \text{ for all } y \in \mathscr{F}_{\boldsymbol{N}}\}.$$

The greedy scheme clearly takes a shortsighted view and tries to maximize the utility that can be obtained from the first arriving call. As can be seen from Fig. 1, the greedy scheme results in lower average utility per unit time than the optimal utility maximizing scheme. We also find that the performance of the greedy scheme deteriorates as we increase the call arrival rate of class-1 calls. This is because the greedy scheme wastes more resources on class-1 calls as λ_1 increases. As remarked in Section 3.3, we find that the performance of the optimal complete partitioning policy is very close to that of the optimal policy. We believe that the optimal complete partitioning policy and the complete sharing policy will complement each other very well, with the complete partitioning policy performing well when the complete sharing policy does not, and vice versa. In this section the numerical examples are meant for the purposes of illustration. It should be noted, however, that our scheme can also be applied to network scenarios including a large number of classes and high capacity values.

5. Pricing mechanism

In this section, we discuss the implications of "selfish behavior" by users and recommend pricing as a means of preventing such behavior. We show that such "selfish behavior" by users results in poor performance of the network, and we develop a pricing mechanism that prevents individual users from deliberately modifying their utility functions. Our resource allocation scheme requires knowledge of the utility functions of all incoming users of the network. The natural way to obtain such information is for each incoming user to communicate to the resource allocator its utility function for the requested connection or call. But this allows a user to deliberately communicate a modified utility function that will optimize its own allocation. Clearly, such a situation is undesirable and should be avoided.

To illustrate the consequences of deliberate modification of the utility functions by users, let us consider the scenario where there are two classes of calls. Let the capacity of the channel be 4 BWUs. Let the utility function of class-1 calls be $[1, 2, 2, 2]$, and let that of class-2 calls be $[0, 3, 3, 3]$. Now let us assume that all class-1 calls communicate their utility function to be $[0, 3, 3, 3]$ while class-2 calls communicate their true utility function. Class-1 users communicate such a modified utility function because they are aware that inflating their utility function in this manner will increase their resource allocation. We set the mean call holding duration of both classes of calls at 60 s. We fix the call arrival rate of class-2 calls at 0.1 calls/s and allow the call arrival rate of class-1 calls to vary between 0.1 and 1 call/s. The resource allocator makes resource allocation decisions assuming that class-1 calls have the utility function

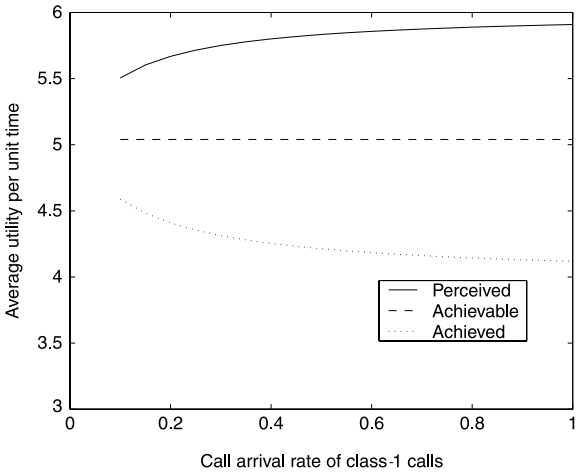

Fig. 2. Consequence of users lying about their utility functions. In this case, class-1 users communicate their utility function as $[0,3,3,3]$ instead of the actual $[1,2,2,2]$. Class-2 users communicate their real utility function $[0,3,3,3]$.

$[0,3,3,3]$. In Fig. 2, we plot the average utility per unit time as a function of the call arrival rate of class-1 calls. We plot the utility the network thinks it is obtaining from its resource allocation decision (perceived utility), the utility that can be achieved if the network knew the actual utility of class-1 calls (achievable utility), and the utility that is actually achieved by using the modified utility function of class-1 users (achieved utility). Clearly, the achieved utility is far below what is achievable. To avoid this situation, there should be disincentives to prevent users from lying about their utility functions. One possibility is to introduce a pricing mechanism that is appropriately designed so that users do not gain from deliberately lying about their utility functions. We now develop such a pricing mechanism.

Let p_k denote the price per unit time that a class-k user is charged. Thus if a class-k user stays in the system for T seconds, then the user is charged $p_k T$ dollars. Such a pricing scheme is *allocation independent* because the price does not depend on the amount of resource allocated. (We will show later that charging users different amounts for different allocations does not result in any additional flexibility in inducing users to reveal their true utility functions.) We also let $d_k(i)$ be the probability that a class-k user is allocated i BWUs. With these notations, the expected rate of return for a class-k user conveying its true utility function $\boldsymbol{U}_k$ can be written as

$$\sum_{i=1}^{C} U_k(i)d_k(i) - p_k,$$

where we assume that the utility functions are also expressed in units of \$/s. On the other hand, if the user were to communicate an alternate utility function $\boldsymbol{U}_l$ instead of its true utility function $\boldsymbol{U}_k$, then the expected reward accrual rate would be

$$\sum_{i=1}^{C} U_k(i)d_l(i) - p_l.$$

In this case, the resource allocator decides the allocation amount and the price to be charged assuming that the user's utility function is $\boldsymbol{U}_l$.

Our objective in using pricing is to ensure that there is no incentive for users to lie about their utility functions. If the prices $\{p_k\}$ are determined in a manner that satisfies the following conditions, then a user with utility function $\boldsymbol{U}_k$ will reveal its true utility function:

$$\sum_{i=1}^{C} U_k(i)d_k(i) - p_k \geqslant \sum_{i=1}^{C} U_k(i)d_l(i) - p_l \quad \text{for all } l \neq k. \tag{11}$$

In other words, if the condition in Eq. (11) is satisfied for all k, then, among all the utility functions that a user can convey, the expected reward rate is highest for its true utility function. On the left hand side of Eq. (11) is the expected reward rate for a class-k user revealing its true utility function, and on the right hand side is the expected reward rate for a class-k user conveying its utility function to be $\boldsymbol{U}_l$. Note that the user's allocation and price are determined by the utility function that the user reveals. Such a pricing mechanism that forces users to reveal their true utility functions is called an *incentive-compatible* pricing scheme [34,35].

The incentive-compatible pricing problem, therefore, is to choose a set of prices $\{p_k\}$ that satisfies Eq. (11) for each $k \in \{1,2,\ldots,M\}$, which is a linear feasibility problem. We have the following necessary condition for the incentive-compatible pricing problem.

$$\sum_{i=1}^{C} (U_k(i) - U_l(i))(d_l(i) - d_k(i)) \leqslant 0$$

for every pair k, l. (12)

The condition in Eq. (12) is obtained by observing that for a class-k user to reveal its true utility function, we need

$$\sum_{i=1}^{C} U_k(i)d_k(i) - p_k \geqslant \sum_{i=1}^{C} U_k(i)d_l(i) - p_l,$$

and for a class-l user to reveal its true utility function we need

$$\sum_{i=1}^{C} U_l(i)d_l(i) - p_l \geqslant \sum_{i=1}^{C} U_l(i)d_k(i) - p_k.$$

Adding these two equations we get the necessary condition in Eq. (12). The necessary condition in Eq. (12) has the interpretation that a class-k user should not be able to gain more by revealing itself to be a class-l user than a class-l user itself gains by revealing its true utility function instead of $\boldsymbol{U}_k$. From the necessary condition in Eq. (12), it follows that an incentive-compatible pricing scheme may not always exist. It can also be shown that the conditions in Eq. (12) are not sufficient for the incentive-compatible pricing problem.

We now show that there is no loss of generality in restricting attention to allocation-independent pricing schemes. Towards this end, we define allocation-dependent pricing schemes and randomized allocation-dependent pricing schemes. A pricing scheme is said to be *allocation dependent* if the price per unit time for a user depends on the amount of resource allocated to it. For the allocation-dependent pricing scheme, we let $p_k(i)$ denote the price per unit time for a class-k user allocated i BWUs. If the per-unit-time price for a user is determined from a probability distribution that depends on the class of the user and the amount of resource allocated, then we call it a randomized allocation-dependent pricing scheme.

Proposition 1. *If there exists an allocation-dependent pricing scheme that guarantees incentive compatibility, then there exists an allocation-independent pricing scheme that guarantees incentive compatibility. Similarly, if there exists a randomized allocation-dependent incentive-compatible pricing scheme, then there exists an allocation-independent incentive-compatible pricing scheme.*

Proof. Let $\{p_k(i)\}$ be the set of allocation-dependent prices that guarantees incentive compatibility. This implies that the following conditions hold for each k:

$$\sum_{i=1}^{C} (U_k(i) - p_k(i))d_k(i) \geqslant \sum_{i=1}^{C} (U_k(i) - p_l(i))d_l(i)$$

for all $l \neq k$. (13)

By choosing the allocation-independent price p_k to be $\sum_{i=1}^{C} p_k(i)d_k(i)$ for each k we see that an allocation-independent price that guarantees incentive compatibility exists whenever there exists an allocation-dependent pricing scheme that guarantees incentive compatibility. Similarly, we can show that if a randomized allocation-dependent incentive-compatible pricing scheme exists, then so does an allocation-independent incentive-compatible pricing scheme. □

The solution to the incentive-compatible pricing problem depends on the values of $d_k(i)$. We can obtain values of $d_k(i)$ by summing the steady-state probabilities of the system being in those states that allocate class-k users i BWUs. This is because the state of the system, as observed by a newly arriving user in steady state, has a probability distribution identical to that of the steady-state distribution of the system. For instance, in the example in Table 1 for the case where $\lambda_1 = 0.5$ and $\lambda_2 = 0.05$, we can obtain the values of $d_1(2)$ and $d_2(2)$ to be 0.0120 and 0.3952, respectively. These values represent the sum of the steady-state probabilities of the system being in states that allocate 2 BWUs to the respective user classes. But, if a group of users decide to collude and lie about their utility functions, these values would be inaccurate because the steady-state distribution is modified as a result of the modification in the call arrival rates. For the example discussed above, to simplify the problem, we make the assumption that $\boldsymbol{U}_1 = [1, 2, 2, 2]$ and $\boldsymbol{U}_2 = [0, 3, 3, 3]$ are the only possible utility functions. Thus users can only

choose one of $\boldsymbol{U}_1$ or $\boldsymbol{U}_2$ as their utility functions. Then, under the no-collusion assumption, it follows from Eq. (11) that any pricing scheme that satisfies the condition $0.7424 \leqslant p_2 - p_1 \leqslant 1.1136$ will be an incentive-compatible scheme, where we have assumed that the units on the utility functions are \$/s.

In addition to the incentive compatibility constraint, it may be desired that a pricing mechanism should ensure that the expected return to users be higher than the price. Without such a condition, users may not be interested in the service at all. This condition can easily be incorporated by imposing the following additional constraint.

$$\sum_{i=1}^{C} U_k(i) d_k(i) \geqslant p_k \quad \text{for each } k. \tag{14}$$

Incorporating the condition in Eq. (14) in the example discussed above, we have that p_1 and p_2 should satisfy the additional conditions that $p_1 \leqslant 0.024$ and $p_2 \leqslant 1.1856$.

6. Conclusions

We developed an optimal resource allocation scheme for elastic applications in multi-class networks. We formulated the resource allocation problem as a continuous-time MDP. In our approach, users specify a utility function that quantifies their benefit corresponding to different resource allocation amounts. Using numerical results, we showed that the optimal scheme results in significant improvement in the average aggregate utility when compared to the greedy resource allocation scheme. To reduce the computational complexity involved in obtaining the optimal scheme, we discussed two types of model reduction techniques. In the first type, the policy obtained is still optimal, and in the second type, the policy would only be approximately optimal but with a higher reduction in the computation complexity. We studied the implications of users lying about their utility functions and developed a pricing mechanism that prevents users from doing so. We also discussed other aspects related to our scheme, such as incorporating constraints on the call blocking probabilities, offline computation, and extension to multiple links and general call holding time distributions.

Acknowledgements

The authors wish to thank Mike Needham and Steve Gilbert of Motorola Labs for useful discussions on the subject of this paper.

References

[1] J. Liebeherr, Multimedia networks: issues and challenges, Computer 28 (4) (1995) 68–69.
[2] S. Shenker, Fundamental design issues for the future Internet, IEEE Journal on Selected Areas in Communications 13 (7) (1995) 1176–1188.
[3] N. Davies, J. Finney, A. Friday, A. Scott, Supporting adaptive video applications in mobile environments, IEEE Communications Magazine 36 (6) (1998) 138–143.
[4] H.R. Varian, Microeconomic Analysis, W.W. Norton, New York, 1992.
[5] F.P. Kelly, A.K. Maulloo, D.K.H. Tan, Rate control for communication networks: shadow prices proportional fairness and stability, Journal of Operations Research Society 49 (3) (1998) 237–252. Available from <http://www.statslab.cam.ac.uk/~frank/PAPERS>.
[6] H. Mine, S. Osaki, Markovian Decision Processes, Elsevier, Amsterdam, 1970.
[7] S.M. Ross, Applied Probability Models with Optimization Applications, Holden-Day, San Francisco, CA, 1970.
[8] M.L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, Wiley, New York, 1994.
[9] D.P. Bertsekas, J.N. Tsitsiklis, Neuro-dynamic Programming, Athena Scientific, Belmont, MA, 1996.
[10] J. Sairamesh, D. Ferguson, Y. Yemini, An approach to pricing, optimal allocation and quality of service provisioning in high-speed packet networks, in: IEEE INFOCOM'95, vol. 3, 1995, pp. 1111–1119.
[11] J.F. Kurose, R. Simha, A microeconomic approach to optimal resource allocation in distributed computer systems, IEEE Transactions on Computers 38 (5) (1989) 705–717.
[12] S.H. Low, D.E. Lapsley, Optimization flow control. I: Basic algorithm and convergence, IEEE/ACM Transactions on Networking 7 (6) (1999) 861–875.
[13] K. Park, M. Sitharam, S. Chen, Quality of service provision in noncooperative networks with diverse user requirements, Decision Support Systems, Special issue on Information and Computation Economics, 28 (2000) 101–122.

[14] H. Yaiche, R.R. Mazumdar, C. Rosenberg, A game theoretic framework for bandwidth allocation and pricing in broadband networks, IEEE/ACM Transactions on Networking 8 (5) (2000) 667–678.
[15] 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.
[16] K.W. Ross, D.H.K. Tsang, Optimal circuit access policies in an ISDN environment: A Markov decision approach, IEEE Transactions on Communications 37 (9) (1989) 934–939.
[17] S. Kalyanasundaram, E.K.P. Chong, N.B. Shroff, Admission control schemes to provide class-level QoS in multiclass networks, in: Proceedings of the 38th Annual Allerton Conference on Control Communication and Computing, vol. 1, 2000, pp. 113–122.
[18] S. Kalyanasundaram, E.K.P. Chong, N.B. Shroff, Admission control schemes to provide class-level QoS in multiclass networks, Computer Networks 35 (2001) 307–326.
[19] E. Altman, Applications of Markov decision processes in communication networks—A survey, Tech. Rep., INRIA, 2000. Available from www.inria.fr/RRRT/RR-3984.html.
[20] I.C. Paschalidis, J.N. Tsitsiklis, Congestion-dependent pricing of network services, IEEE/ACM Transactions on Networking 8 (2) (2000) 171–184.
[21] G. de Veciana, C. Courcoubetis, J. Walrand, Decoupling bandwidths: A decomposition approach to resource management in networks, in: IEEE INFOCOM'94, vol. 2, Toronto, Canada, 1994, pp. 446–474. Available from http://walrandpc.eecs.berkeley.edu/Pubs.html.
[22] K.W. Ross, Randomized and past-dependent policies for Markov decision processes with multiple constraints, Operations Research 37 (3) (1989) 474–477.
[23] R. Givan, S. Leach, T. Dean, Bounded parameter Markov decision processes, in: Proceedings of the Fourth European Conference on Planning, 1997, pp. 234–246. Available from <http://dynamo.ecn.purdue.edu/~givan>.
[24] J.K. Satia, R.E. Lave Jr., Markovian decision processes with uncertain transition probabilities, Operations Research 21 (1973) 728–740.
[25] S. Kalyanasundaram, E.K.P. Chong, N.B. Shroff, Markov decision processes with uncertain transition rates: Sensitivity and robustness, Tech. Rep., Purdue University, 1999.
[26] T. Dean, R. Givan, S. Leach, Model reduction techniques for computing approximately optimal solutions for Markov decision processes, in: Uncertainty in AI, 1997. 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. Available from <http://dynamo.ecn.purdue.edu/~givan>.
[28] P. Marbach, O. Mihatsch, J.N. Tsitsiklis, Call admission control and routing in intergrated service networks using neuro-dynamic programming, IEEE Journal on Selected Areas in Communications 18 (2) (2000) 197–208.
[29] K.W. Ross, D.H.K. Tsang, The stochastic knapsack problem, IEEE Transactions on Communications 37 (7) (1989) 740–747.
[30] 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.
[31] J.S. Kaufman, Blocking in a shared resource environment, IEEE Transactions on Communications 29 (10) (1981) 1474–1481.
[32] 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.
[33] A. Pfening, M. Telek, Optimal control of Markov regenerative processes, IEEE International Conference on Systems Man and Cybernetics 1 (1998) 663–668.
[34] R.M. Bradford, Pricing, routing, and incentive compatibility in multiserver queues, European Journal of Operational Research 89 (2) (1996) 226–236.
[35] Y.A. Korilis, A. Orda, Incentive compatible pricing strategies for QoS routing, in: IEEE INFOCOM'99, vol. 2, New York, March 1999, pp. 891–899.

Suresh Kalyanasundaram received his bachelor 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 for Motorola. His research interests are in the area of performance analysis of wired and wireless networks.

Edwin K.P. Chong 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 joined the School of Electrical and Computer Engineering at Purdue University in 1991, where he was named a University Faculty Scholar in 1999, and was promoted to Professor in 2001. Since August 2001, he has been a Professor of Electrical and Computer Engineering at Colorado State University. His current interests are in communication networks and optimization methods. He co-authored the recent book, An Introduction to Optimization, 2nd Edition, Wiley-Interscience, 2001. He was on the editorial board of the IEEE Transactions on Automatic Control, and is currently an editor for Computer Networks. He is an IEEE Control Systems Society distinguished lecturer. He received the NSF Career Award in 1995 and the ASEE Frederick Emmons Terman Award in 1998.

Ness B. Shroff received his Ph.D. degree from Columbia University, NY, in 1994. He is currently an associate professor in the School of Electrical and Computer Engineering at Purdue University. His research interests span the areas of wireless and wireline communication networks. He is especially interested in fundamental problems involving the design, performance, scheduling, pricing, and control of these networks. His research is funded by numerous companies and federal and state agencies. Dr. Shroff is an editor for IEEE/ACM Transactions on Networking and Computer Networks, and past editor of IEEE Communications Letters. He was the conference chair for the 14th Annual IEEE Computer Communications Workshop (in Estes Park, CO, Oct. 1999) and program co-chair for the Symposium on High-Speed Networks, Globecom 2001 (San Francisco, CA, Nov. 2000). He is also the technical program co-chair for IEEE INFOCOM'03. He received the NSF Career award in 1996.