

# On-line Tuning of Prices for Network Services

Enrique Campos-Náñez and Stephen D. Patek  
Department of Systems and Information Engineering  
University of Virginia  
Charlottesville, Virginia 22904  
Email: {ecamposn,patek}@virginia.edu

**Abstract**—Recent investigations into the pricing of multiclass loss networks have shown that static prices are optimal in the asymptotic regime of many small sources. These results suggest that nearly optimal prices for highly aggregated systems can be computed from the solution to a limiting deterministic optimization model. When the assumption of many small sources does not hold, static prices are still preferable (for practical reasons), but we are left with the difficult issue of computing an optimal solution when the stochastic nature of the process cannot be ignored. In this paper, we develop a computational procedure for optimizing static prices that operates by adjusting prices in response to actual customer arrivals and departures and is robust to parametric uncertainty about the underlying system. We provide initial arguments for the convergence properties of our optimization algorithm, and we illustrate its application in several numerical examples.

## I. INTRODUCTION

The explosive growth of Internet-based industries in the last decade and the recent proliferation of corporate networks raises the question of how to accommodate the ever growing demand for network services. Often, especially in the core of the Internet, the answer is to “over engineer” the system by installing more capacity when the need arises. [1] However, in many contexts, such as in broadband access networks, the option to add extra capacity is not so clear, potentially making it necessary to introduce arbitration mechanisms that efficiently allocate resources and regulate network traffic.

Pricing has been studied as one such mechanism. In [2], prices are defined on a per-unit-flow basis and are interchanged between a set of distributed regulating entities to achieve fairness. In [3], prices per unit flow are calculated to optimize a network objective for differentiated services. Similar approaches can be found in [4]–[8], where usage-based charging is achieved by communicating resource prices (which can be interpreted as Lagrangian multipliers for network capacity constraints) between the network regulating entities. In [9], [10], a traditional economic pricing mechanism is used by the (central) owner of the network resources to establish the fees to be charged per unit of each resource involved in a service request. A similar approach is used in [11] where the length of the queue at a given network link is used to set prices for network resources, and to achieve congestion control. Game theoretic approaches, that use market mechanisms, such as bidding, to implement pricing mechanisms that work in a distributed manner to achieve a fair use of the resources have been studied in [3], [12]–[14]. A client-based approach is

developed in [15]–[17], where prices for a given priority of service are fixed beforehand, but the user (flow) has to decide how much of its flow to send using each priority class. The effect of best-effort traffic has been studied in [18]–[23], and more recently in [24].

Per-session pricing models in multiclass loss networks have been considered in [25]–[28], where the resource requirements for accepted calls in each class are reserved by some mechanism, such as RSVP, and the issue is to characterize pricing policies that maximize overall revenue or social welfare. In [27] (for the single link case) and [28] (for the network case), the near-optimality of static pricing policies has been established. Similar results appear in [23] under more general service time distributions.

In this paper, we focus on the single link case and address computational issues associated with the per-session pricing model in [27]. Our goal is to develop “on-line” computational algorithms that apply while the network is in operation. So far, the literature on optimal per-session pricing has focused on off-line computation, where the parameters for the multiclass loss network are given in advance and the main issue is to determine an optimal set of prices assuming that the given parameter values are correct. Specifically, in [27], Paschalidis and Tsitsiklis compute dynamic pricing policies for a single link model by means of both exact and approximate dynamic programming, and they compare these results to the optimal static prices obtained from the solution to a deterministic optimization model that applies in the many small sources asymptotic regime. Later, in [28], Paschalidis and Liu adapt the simulation-based algorithm of [29] to compute optimal static prices for the network case (where dynamic programming can be ruled out due to the so-called “curse of dimensionality”).

The main drawback to the computational procedures from [27], [28] is that they do not naturally extend to an on-line setting, where (1) the only model available for the process is the process itself and (2) robustness to unknown or slowly time-varying parameters (e.g. arrival rates and holding times) is critical. This paper explores the possibility of adapting the simulation-based procedure of [30] (based on [29]) to allow for efficient and robust on-line pricing of network services. Our first contribution is to spell out a version of the simulation-based procedure from [30] that applies to the continuous-time operation of the system and, in the case where arrival rate functions and expected holding times are known, is guaranteed to converge with probability one to an extremum of the average

reward process. Next, to address the case where the parameters of the model are unknown (and possibly slowly time varying), we develop an adaptive price-setting algorithm by allowing for the simultaneous estimation of model parameters. Our numerical results show that the adaptive algorithm has the ability to identify nearly-optimal prices, even with significant uncertainty about arrival rates and holding times.

The remainder of this paper is organized as follows. In Section II, we review the optimization model from [27] for pricing services in multiclass loss systems with a single resource. We focus on the problem of revenue maximization with respect to static prices. In Section III, toward the goal of developing an on-line pricing mechanism, we discuss a model-based optimization procedure (Algorithm 1) adapted from the simulation-based algorithm of [29]. Even though this algorithm operates by adjusting prices in response to arrivals and departures in real-time, we refer to it as “model-based” because it requires explicit knowledge of the parameters of the underlying system. (The main difference between Algorithm 1 and the earlier approach from [29] is that Algorithm 1 incorporates a device known as  $i^*$ -adaptation [30] which resolves an important technical issue associated with the original simulation-based procedure.) In Section IV, we propose an on-line algorithm (Algorithm 2) which is adapted from the model-based procedure of Section III and allows for the simultaneous estimation of model parameters. In Section V, we present a numerical investigation of Algorithms 1 and 2, focusing particularly on the ability of Algorithm 2 to track changes in the parameters of the underlying model. In Section VI, we summarize the paper and outline directions for future research.

## II. PROBLEM FORMULATION

We consider a multiclass loss system with a single resource of capacity  $C$  shared by users from  $K$  classes of calls or services.<sup>1</sup> Each service class  $k \in \{1, 2, \dots, K\}$  can be described by an arrival rate function  $\alpha_k(u_k)$ , which we assume to be a non-increasing twice differentiable function of a price  $u_k \in [0, \bar{u}_k]$ , with bounded first and second derivatives. We assume that requests for calls constitute a Poisson arrival process with rate  $\alpha_k(u_k)$  whenever price  $u_k$  is used. To disallow the possibility that we “turn off” the arrival process associated with any class, we assume that  $\alpha_k(\bar{u}_k) > 0$  for all  $k = \{1, 2, \dots, K\}$ . Let  $\mathcal{U} = \prod_{k=1}^K [0, \bar{u}_k]$  denote the set of all feasible price combinations. The duration of a call of class  $k$  is assumed to be exponentially distributed with mean  $1/\beta_k$ , during which the network reserves  $m_k$  out of the total capacity  $C$ . At a time  $t$ , the state of the system can be described as  $i(t) = (i_1(t), i_2(t), \dots, i_K(t))$ , a vector whose  $k$ -th element represents the number of ongoing calls of class  $k$  present in the system. We use  $I$  to denote the set of all feasible usage profiles  $(i_1, i_2, \dots, i_K)$ . (The assumption of Poisson arrivals and exponential service is common in the literature on network pricing. For us, the fact that the system is memoryless is

<sup>1</sup>As an example, consider a broadband access network with a single uplink to Internet.

critical, in motivating the computational techniques that we develop.)

Finally, we assume that the system implements a strict admission control rule. Suppose that at time  $t$  a request for a call of type  $k$  is received. The call will be admitted if  $m \cdot i(t) + m_k \leq C$ , where the vector  $m$  is defined by  $m = (m_1, m_2, \dots, m_K)$ . To simplify notation, define  $A(i) = \{k | m \cdot i + m_k \leq C\}$  to be the set of calls that satisfy the admission control rule when the system is in state  $i \in I$ .

In this paper, we focus on the problem of setting prices to shape the arrival process so that we maximize the expected rate at which revenue accrues to the service provider. Define  $\Pi$  to be the set of mappings  $\pi : I \rightarrow \mathcal{U}$  that assign a set of prices to each state  $i \in I$ . The revenue maximization problem is then defined as

$$\max_{\pi \in \Pi} \lim_{T \rightarrow \infty} \frac{1}{T} E \left[ \int_{t=0}^T \sum_{k \in A(i(t))} \alpha_k(\pi_k(i(t))) \pi_k(i(t)) dt \right].$$

As discussed in [27], there is no dependence of the average reward on the initial state  $i(0)$ , and we use  $\lambda^*$  to denote the optimal value of the problem. We point out that the same modeling framework can be used to capture the problem of maximizing social welfare, and both the revenue maximization and social welfare objectives are of the average reward form, and can be solved using traditional dynamic programming techniques (see, for example, [31]).

### A. Static Pricing Policies

For simplicity, we restrict our attention to a subset of  $\Pi$ , namely the so-called static pricing policies, where, for a given solution, the same class-dependent price  $u_k$  always applies regardless of the state of the process. The revenue maximization problem then can be expressed as  $\max_{u \in \mathcal{U}} \lambda(u)$ , where

$$\lambda(u) = \lim_{T \rightarrow \infty} \frac{1}{T} E \left[ \int_0^T \sum_{k \in A(i(t))} \alpha_k(u_k) u_k dt \right]. \quad (1)$$

### B. Discussion of Previous Results

A theoretical justification for restricting attention to static pricing policies is given in [27], where static prices are shown to be optimal in the asymptotic regime of “many small sources.” Specifically, consider a scaled version of the system which is obtained by scaling the original capacity of the link as  $C^{(c)} = cC$ , where  $c > 1$ , and by scaling the original arrival rate functions as  $\alpha_k^{(c)}(u_k) = c\alpha_k(u_k)$ . Let  $\lambda_c^*$  be the average reward obtained by an optimal dynamic policy, and let  $\lambda(u_c^*)$  be the average reward associated with an optimal static policy. A key result from [27] is that  $\lim_{c \rightarrow \infty} \lambda_c^* = \lim_{c \rightarrow \infty} \lambda(u_c^*)$ , i.e. that static pricing policies are optimal in the asymptotic regime. Consequently, nearly optimal static prices exist for cases where the bandwidth requirements  $m$  are small compared to the capacity  $C$ . This result has been extended in [23] to general service distributions.

While the near-optimality of static pricing policies is reassuring, if the assumption of many small users does not

apply (or is questionable), the solution to the deterministic optimization model that gives rise to the upper bound in [23], [27], [28] may be far from optimal, and we must deal with the practical question of how to compute optimal static prices.

### C. On-line Pricing Algorithms

Our goal in this paper is to make progress toward “on-line” pricing algorithms, where static prices for services are adjusted in response to observations of the system in operation. We seek to achieve the following characteristics in our computational approach.

- *Real-Time Operation:* Our price optimization mechanism should operate with respect to observations of the actual system in operation. That is, rather than assuming access to a simulation of the process (which could be used for off-line calculations) we only require access to the real-time stream of arrivals and departures of calls of various classes.
- *Efficient Operation:* Our computational algorithm should have no need to store per-call information and should maintain only  $O(K)$  state variables, where  $K$  is the number of classes of calls.
- *Robust Operation:* Our optimization procedure should have the ability to discover and respond to uncertain and possibly time-varying parameters of the underlying system.

In Section III below, we develop a model-based on-line price-setting algorithm which achieves the first two characteristics above. We refer to this algorithm as “model-based” because it cannot be implemented without knowledge of the parameters of the model, particularly the arrival rate functions  $\alpha_k(u_k)$  and the expected holding times  $1/\beta_k$ . In Section IV, we extend the model-based algorithm so that it can adapt to unknown and slowly time-varying parameters, thereby addressing the third characteristic above.

## III. MODEL-BASED PRICING

The pricing model of Section II can be viewed as a continuous-time Markov reward process, where (in the case of revenue maximization) the objective is to maximize the expected average rate at which reward accrues to the service provider. As is done in [27], [28], it is possible to uniformize the process (see, for example, Chapter 5 of [31]) to obtain an equivalent discrete-time description of the process. Conceptually, the uniformization involves Poisson sampling of the process at a rate  $\nu^*$  equal to (or higher than) the fastest possible transition rate of the continuous-time system. For the pricing model of Section II, the rate of transitions out of state  $i \in I$  is

$$\nu_i(u) = \sum_{k \in A(i)} \alpha_k(u_k) + \sum_{k=1}^K i_k \beta_k, \quad (2)$$

from which we obtain the following upper bound on the rate of transitions

$$\nu^* = \sum_{k=1}^K \lfloor C/m_k \rfloor \beta_k + \sum_{k=1}^K \alpha_k(0). \quad (3)$$

With  $\nu^*$  in hand, we may infer discrete-time state transition probabilities, as well as expected rewards associated with accepted call arrivals, and, in principle, we may apply any algorithmic technique available to us to optimize the average-reward of discrete-time Markov processes. In this section, we adapt the discrete-time simulation-based methodology of [29] (as extended in [30], [32]) for on-line use in a continuous-time setting.

### A. An Algorithm to Compute Optimal Prices

The rationale behind the discrete-time algorithm of [29] is (1) to exploit the regenerative structure of the underlying Markov process to compute asymptotically unbiased estimates of the gradient of the objective function and (2) to use the estimates of the gradient in adjusting the control parameters (i.e. prices) toward an extremum of the average reward function. An important feature of the methodology in [29] is that updates to the controllable parameters occur only when the system transitions into a specially marked recurrent state  $i^* \in I$ . Unfortunately, for a given set of control parameters, transitions to  $i^*$  can be infrequent, resulting in high-variance gradient estimates and ultimately in unacceptably slow convergence. This issue has been addressed in [30] through the introduction of an  $i^*$ -adaptation procedure, where the marked state  $i^*$  can be reset as needed to account for changes in the steady-state distribution of the process as the control parameters are adjusted.

In Algorithm 1 below, we present a price-setting algorithm, which can be interpreted as a continuous-time implementation of the discrete-time algorithm in [30]. Since the algorithm proceeds in response to arrival and departure events of the actual process, it applies in an on-line setting.

#### Algorithm 1 - Model-Based Price Optimization

Given scale parameter  $\eta > 0$ , stepsize rule  $\{\gamma_m : m = 1, 2, \dots\}$  such that  $\sum_m \gamma_m = \infty$ , and  $\sum_m \gamma_m^2 < \infty$ , initial observation period threshold  $\tau > 0$ , initial set of prices  $(u_{0,1}, u_{0,2}, \dots, u_{0,K})$  for service classes  $1, \dots, K$ , and an estimate  $\tilde{\lambda}$  of the average reward associated with the initial set of prices, compute recursively  $\{(u_m, \tilde{\lambda}_m, \tau_m, i_m^*, s_m) : m = 0, 1, \dots\}$  as follows.

(Initialization)

Set  $m = 0$ ,  $u_0 = (u_{0,1}, u_{0,2}, \dots, u_{0,K})$ ,  $\tilde{\lambda}_0 = \tilde{\lambda}$ , and  $\tau_0 = \tau$ . Set  $t_0$  to be the time at which the algorithm is initialized, and set  $i_0^*$  to be the state of the system at time  $t_0$ . Set the initial timeout threshold  $s_0 = t_0 + \tau_0$ . Finally, set local variables  $F = \mathbf{0} \in \mathbb{R}^K$ ,  $L = \mathbf{0} \in \mathbb{R}^K$ , and  $G = 0$ .

( $n$ -th State Transition)

Let  $t_n$  denote the time of the  $n$ -th state transition, and let  $i_n = i(t_n^+)$  be the new state of the system at that time. Proceed as follows.

- 1) (Update Local Variables) First, generate a Poisson random variable  $T$  as

$$T \sim \text{Poisson}((t_n - t_{n-1})(\nu^* - \nu_{i(t_{n-1})}(u_m))), \quad (4)$$

where the argument in the right hand side is the mean of  $T$ . Next, update  $F$ ,  $L$ , and  $G$  as follows.

$$F = F + T \cdot L \left( g_{i_{n-1}}(u_m) - \tilde{\lambda}_m \right) \quad (5)$$

$$- \frac{T^2 + T}{2} (g_{i_{n-1}}(u_m) - \tilde{\lambda}_m) l_{i_{n-1}}(u_m) \quad (6)$$

$$+ T \nabla g_{i_{n-1}}(u_m), \quad (7)$$

$$L = L + T l_{i_{n-1}}(u_m), \text{ and} \quad (8)$$

$$G = G + T (g_{i_{n-1}}(u_m) - \tilde{\lambda}_m), \quad (9)$$

where  $g_i(u) = \sum_{k \in A(i)} \alpha_k(u_k) u_k / \nu^*$ , and  $l_i(u) = \sum_{k \in A(i)} \frac{-\alpha'_k(u_k)}{\nu^*} e_k$ , define  $e_k$  as the  $k$ -th unit vector in  $\mathbb{R}^K$ . In addition, if the  $n$ -th state transition corresponded to a class- $k$  arrival, then update  $L$  again as

$$L = L + \sum_{k \in A(i_{n-1})} \frac{\alpha'_k(u_{m,k})}{\alpha_k(u_{m,k})} e_k,$$

and use  $L$  to revise  $F$  and  $G$  as

$$F = F + T \cdot L \left( g_{i_n}(u_m) - \tilde{\lambda}_m \right) + \nabla g_{i_n}(u_m),$$

$$G = G + T (g_{i_n}(u_m) - \tilde{\lambda}_m).$$

- 2) (Regenerative Cycle Complete) If the  $n$ -th state transition resulted in a return to the state  $i_m^*$ , then update the price vector and the average reward estimate as

$$u_{m+1} = u_m + \gamma_m F, \quad (10)$$

$$\tilde{\lambda}_{m+1} = \tilde{\lambda}_m + \eta \gamma_m G. \quad (11)$$

Leave the observation period threshold and marked state unchanged as  $\tau_{m+1} = \tau_m$  and  $i_{m+1}^* = i_m^*$ . Set a new timeout threshold  $s_{m+1} = t_n + \tau_{m+1}$ . Finally, reset the local variables  $F = \mathbf{0}$ ,  $L = \mathbf{0}$ , and  $G = 0$ , and set  $m = m + 1$ .

(Timeout)

If real time  $t$  ever reaches the current timeout threshold  $s_m$ , then proceed as follows.

Increase the observation period threshold as  $\tau_{m+1} = \tau_m + 1/\nu^*$ . Leave the price vector and average reward estimate unchanged as

$$\begin{aligned} u_{m+1} &= u_m, \\ \tilde{\lambda}_{m+1} &= \tilde{\lambda}_m. \end{aligned}$$

Set  $i_{m+1}^* = i(t)$  and  $s_{m+1} = t + \tau_{m+1}$ . Finally, reset the local variables as  $F = \mathbf{0}$ ,  $L = \mathbf{0}$ , and  $G = 0$ , and set  $m = m + 1$ .

## B. Discussion

Algorithm 1 works by collecting reward and likelihood information during regenerative cycles of the process to obtain asymptotically unbiased estimates of the gradient of the objective function. The ‘‘local’’ variables,  $F$ ,  $L$ , and  $G$ , all have interpretations similar to those in [29], [30]. Specifically, at the end of a regenerative cycle, the vector  $F$  contains an estimate

of the gradient of the objective function, and this estimate is used in Eqn. (10) to update the set of prices for services. The vector  $L$  contains likelihood information about the observed system trajectory, and the scalar  $G$  contains information about the objective function and is used in Eqn. (11) to improve the estimate  $\tilde{\lambda}_m$ . Note the small memory footprint of this algorithm: we need only store and manipulate  $(3K + 5)$  real variables as the optimization procedure evolves.

As mentioned earlier, Algorithm 1 can be interpreted as a continuous-time version of the optimization procedure of [29], [30], which was originally developed for discrete-time Markov chains. The conversion from discrete-time to continuous time here is mostly straightforward, although there is one peculiar feature of the algorithm that arises from the required uniformization of the Markov chain. From uniformization, the equivalent discrete-time Markov system introduces artificial ‘‘self-transition’’ events that figure significantly in the evolution of the optimization algorithm. The continuous-time implementation of this algorithm observes only actual system changes, and hence must provide a mechanism to correct the absence of self-transition events. Recall that the constants  $\nu^*$  and  $\nu_i(u)$  are, respectively, an upper bound on the transition rate of the Markov chain and the transition rate out of state  $i$  when the price vector  $u$  is used. Recall also that the equivalent discrete-time model of the continuous-time process is constructed from Poisson sampling at rate  $\nu^*$ . Thus,  $T$  in Eqn. (4) corresponds to the random number of self transitions which occur at a rate of  $[\nu^* - \nu_{i(t_{n-1})}(u_m)]$  while the system remains at state  $i(t_{n-1})$ . The information related to self-transition events can be considered by generating a random number of these events at the rate described in Eqn. (4), and accounting for the lost information in Eqns. (5)-(9).

The subroutine ‘‘(Timeout)’’ provides a continuous-time implementation of the  $i^*$ -adaptation procedure from [30], which serves to prevent the optimization algorithm from getting stuck in lengthy regenerative cycles. To recognize the need for such a feature recall that updates to the price vector occur only at renewal points (i.e. upon transitions to the specially marked recurrent state  $i^* \in I$ ). As the optimization algorithm progresses and an optimal set of prices emerges, the set of states that are likely to be visited frequently can vary significantly, and it can be difficult to choose a priori a fixed state to serve as  $i^*$  throughout the entire evolution of the process. The  $i^*$ -adaptation procedure avoids this difficulty by setting an observation period threshold  $\tau$  that is enforced in the sense that only regenerative cycles shorter than  $\tau$  are allowed to impact the evolution of prices (i.e. regenerative cycles longer than  $\tau$  are ignored). As discussed in [30], to avoid biasing the results, it is critical for the threshold  $\tau$  to be incremented over time so that, asymptotically, no regenerative cycles get dropped. In present context, the  $i^*$ -adaptation procedure is implemented as a timeout process, where the current observation period threshold  $\tau_m$  is used to set a timeout epoch  $s_m$  at which time the marked state will be reset to a better value (specifically, the current state  $i_n$ ), and the data collected up until the timeout period is ignored.

*Theorem 1 (Convergence of Algorithm 1):* Given the step-size rule  $\gamma_m = a/(b + m)$ , with  $a, b > 0$ , and assuming that the model parameters  $\alpha_k(\cdot)$  and  $\beta_k$  are known for all service classes  $k = 1, \dots, K$ , Algorithm 1 converges with probability one to critical point  $u^*$  where  $\nabla\lambda(u^*) = 0$ .

This theorem follows primarily from the main result of [30] since, under the modeling assumptions of Section II, all states of the process are recurrent. One small technical issue, resolved in the companion technical report [33], deals with the fact that Algorithm 1 replaces the discrete-time timeout threshold of the  $i^*$ -adaptation algorithm with a fixed timeout interval in the continuous-time implementation. We illustrate Algorithm 1 in action in Section V after first discussing an adaptive version of the algorithm, which is designed to be robust to parametric uncertainty about the process.

#### IV. ADAPTIVE PRICING

One of the shortcomings of the algorithm of the preceding section is the assumption of knowledge of all the parameters involved in the model. In this section, we extend the model-based algorithm to allow for the simultaneous estimation of model parameters and provide initial arguments for the convergence properties of the resulting adaptive price-setting algorithm.

To set the stage for this discussion, we assume that arrival rate functions  $\alpha_k(\cdot)$  and service rates  $\beta_k$  are parameterized by a set of fixed but unknown parameters  $\theta^* \in \mathbb{R}^q$ , which is assumed to belong to a compact set  $\Theta \subset \mathbb{R}^q$ . As before, we assume that requests for service arrive according to independent Poisson processes with rates  $\alpha_1(u_1, \theta), \dots, \alpha_K(u_K, \theta)$  and that service times are exponentially distributed with rates  $\beta_1(\theta), \dots, \beta_K(\theta)$ . We assume that all of the parameters  $\alpha_k$  and  $\beta_k$  are twice differentiable in  $(u, \theta)$ , with bounded first and second derivatives. Analogously to the preceding section, we define

$$\nu_i(u, \tilde{\theta}) = \sum_{k \in A(i)} \alpha_k(u_k, \tilde{\theta}) + i_k \beta(\tilde{\theta}),$$

and we assume knowledge of a uniform bound  $\nu^*$  on the transition rate of the underlying process, i.e.

$$\nu^* \geq \nu_i(u, \theta), \quad \forall u \in \mathcal{U}, \theta \in \Theta, i \in I.$$

Our objective from here is to create an efficient and robust on-line procedure for optimizing static prices  $u \in \mathcal{U}$  while at the same time estimating the parameters  $\theta^* \in \Theta$ .

#### Algorithm 2 - Adaptive Price Optimization

Given scale parameters  $\eta > 0$  and  $\mu > 0$ , stepsize rule  $\{\gamma_m : m = 1, 2, \dots\}$  such that  $\sum_m \gamma_m = \infty$ , and  $\sum_m \gamma_m^2 < \infty$ , initial observation period threshold  $\tau > 0$ , initial set of prices  $(u_{0,1}, u_{0,2}, \dots, u_{0,K})$  for service classes  $1, \dots, K$ , an estimate  $\tilde{\lambda}$  of the average reward associated with the initial set of prices, and an initial estimate  $\tilde{\theta}_0$  of the unknown parameters  $\theta^*$ , compute recursively  $\{(u_m, \tilde{\lambda}_m, \tilde{\theta}_m, \tau_m, i_m^*, s_m) : m = 0, 1, \dots\}$  as follows.

(Initialization)

Set  $m = 0$ ,  $u_0 = (u_{0,1}, u_{0,2}, \dots, u_{0,K})$ ,  $\tilde{\lambda}_0 = \tilde{\lambda}$ ,  $\tilde{\theta}_0 = \tilde{\theta}$ , and  $\tau_0 = \tau$ . Set  $t_0$  to be the time at which the algorithm is initialized, and set  $i_0^*$  to be the state of the system at time  $t_0$ . Set the initial timeout threshold  $s_0 = t_0 + \tau_0$ . Finally, set local variables  $F = \mathbf{0} \in \mathbb{R}^K$ ,  $L = \mathbf{0} \in \mathbb{R}^K$ ,  $\mathcal{L} = \mathbf{0} \in \mathbb{R}^K$ , and  $G = 0$ .

( $n$ -th State Transition)

Let  $t_n$  denote the time of the  $n$ -th state transition, and let  $i_n = i(t_n^+)$  be the new state of the system at that time. Proceed as follows.

- 1) (Update Local Variables) First, generate a Poisson random variable  $T$  according to

$$T \sim \text{Poisson}((t_n - t_{n-1})(\nu^* - \nu_{i_n}(u_m, \tilde{\theta}_m))).$$

Next, update  $F, L, \mathcal{L}$  and  $G$  as follows.

$$\begin{aligned} F &= F + T \cdot L \left( g_{i_{n-1}}(u_m, \tilde{\theta}_m) - \tilde{\lambda}_m \right) \\ &\quad - \left[ \frac{T(T+1)}{2} (g_{i_{n-1}}(u_m, \tilde{\theta}_m) - \tilde{\lambda}_m) \right. \\ &\quad \left. \cdot l_{i_{n-1}}(u_m, \tilde{\theta}_m) \right] \\ &\quad + T \nabla g_{i_{n-1}}(u_m, \tilde{\theta}_m), \\ L &= L + T \cdot l_{i_{n-1}}(u_m, \tilde{\theta}_m), \\ \mathcal{L} &= \mathcal{L} - T \frac{\nabla_{\tilde{\theta}_m} \nu_{i_{n-1}}(u_m, \tilde{\theta}_m)}{\nu^* - \nu_{i_{n-1}}(u_m, \tilde{\theta}_m)}, \\ G &= G + T (g_{i_{n-1}}(u_m, \tilde{\theta}_m) - \tilde{\lambda}_m). \end{aligned}$$

where  $g_i(u, \tilde{\theta}) = \sum_{k \in A(i)} \alpha_k(u_k, \tilde{\theta}) u_k / \nu^*$ , and  $l_i(u, \tilde{\theta}) = \sum_{k \in A(i)} \frac{-\alpha'_k(u_k, \tilde{\theta})}{\nu^*} e_k$ . In addition, if the  $n$ -th state transition corresponded to a class- $k$  arrival, then update  $L$  and  $\mathcal{L}$  again as

$$\begin{aligned} L &= L + \sum_{k \in A(i_{n-1})} e_k \frac{\alpha'_k(u_{m,k}, \tilde{\theta}_m)}{\alpha_k(u_{m,k}, \tilde{\theta}_m)}, \\ \mathcal{L} &= \mathcal{L} + \frac{\nabla_{\tilde{\theta}_m} \alpha_k(u_m, \tilde{\theta}_m)}{\alpha_k(u_m, \tilde{\theta}_m)}. \end{aligned}$$

Otherwise, if the  $n$ -th state transition corresponded to a class- $k$  departure, then update  $\mathcal{L}$  again as

$$\mathcal{L} = \mathcal{L} + \frac{\nabla_{\tilde{\theta}_m} \beta_k(\tilde{\theta}_m)}{\beta_k(\tilde{\theta}_m)}$$

(Here, the derivative is taken with respect to the estimate of the parameters  $\tilde{\theta}$ ). Finally, use the result of the event-specific calculations above to make final adjustments to  $F$  and  $G$  as follows.

$$\begin{aligned} F &= F + T \cdot L \left( g_{i_n}(u_m, \tilde{\theta}_m) - \tilde{\lambda}_m \right) \\ &\quad + \nabla g_{i_n}(u_m, \tilde{\theta}_m), \\ G &= G + T (g_{i_n}(u_m, \tilde{\theta}_m) - \tilde{\lambda}_m). \end{aligned}$$

2) (Regenerative Cycle Complete) *If the  $n$ -th state transition resulted in a return to the state  $i_m^*$ , then update the price vector, the average reward estimate, and the parameter vector estimate as*

$$\begin{aligned} u_{m+1} &= u_m + \gamma_m F, \\ \tilde{\lambda}_{m+1} &= \tilde{\lambda}_m + \eta \gamma_m G, \\ \tilde{\theta}_{m+1} &= \tilde{\theta}_m + \mu \gamma_m \mathcal{L}. \end{aligned} \quad (12)$$

*Leave the observation period threshold and marked state unchanged as  $\tau_{m+1} = \tau_m$  and  $i_{m+1}^* = i_m^*$ . Set a new timeout threshold  $s_{m+1} = t_n + \tau_{m+1}$ . Finally, reset the local variables  $F = \mathbf{0}$ ,  $L = \mathbf{0}$ ,  $\mathcal{L} = \mathbf{0}$ , and  $G = 0$ , and set  $m = m + 1$ .*

(Timeout)

*If real time  $t$  ever reaches the current timeout threshold  $s_m$ , then proceed as follows.*

*Increase the observation period threshold as  $\tau_{m+1} = \tau_m + 1/\nu^*$ . Leave the price vector, the average reward estimate, and the parameter vector estimate unchanged as*

$$\begin{aligned} u_{m+1} &= u_m, \\ \tilde{\lambda}_{m+1} &= \tilde{\lambda}_m, \\ \tilde{\theta}_{m+1} &= \tilde{\theta}_m. \end{aligned} \quad (13)$$

*Set  $i_{m+1}^* = i(t)$  and  $s_{m+1} = t + \tau_{m+1}$ . Finally, reset the local variables as  $F = \mathbf{0}$ ,  $L = \mathbf{0}$ ,  $\mathcal{L} = \mathbf{0}$ , and  $G = 0$ , and set  $m = m + 1$ .*

#### A. Discussion

As an extension of the model-based algorithm of the preceding section, Algorithm 2 works by collecting reward and likelihood information during regenerative cycles of the process to obtain estimates of the gradient of the objective function. All of the local variables from the model-based algorithm show up here, with the addition of  $\mathcal{L}$  which collects likelihood information used in Eqn. (12) to improve estimates of the unknown parameters  $\theta^*$ . Note that, while Algorithm 2 uses the estimates  $\tilde{\theta}_m$  to perform the same optimization procedure described in Algorithm 1, it simultaneously uses observed system behavior to further improve the estimates  $\tilde{\theta}_m$ . In this sense, Algorithm 2 is similar in spirit to earlier algorithms developed for the adaptive control of Markov chains (see, for example, [34]). Again, note the small memory footprint of the algorithm: we need only store and manipulate  $(3K + 2q + 5)$  real variables as the optimization procedure evolves, where  $q$  is the dimension of the parameter vector  $\tilde{\theta}$  and  $K$  is the number of the service classes (as usual).

The process by which Algorithm 2 estimates the parameter vector  $\theta^*$  [cf. Eqns. (12-13)] has the following rationale. Suppose a Markov chain evolves according to  $P_{ij}(\theta^*) = P(i_{n+1} = j \mid i_n = i, \theta^*)$ , with  $\theta^*$  unknown. Upon observing the state trajectory  $\{i_0, i_1, \dots\}_2$ , we could estimate  $\theta^*$  by identifying the parameter vector  $\theta^*$  that maximizes the

likelihood of observing  $\{i_0, i_1, \dots\}$ . We instantiate this idea by updating  $\tilde{\theta}$  in the direction of the gradient of the likelihood associated with each observed regenerative cycle. Specifically, given an observed regenerative cycle  $\{i_0, i_1, \dots, i_T\}$  in the uniformized discrete-time process, the likelihood of observing this particular trajectory under  $\tilde{\theta}_m$  and the prevailing set of prices  $u_m$  is given by

$$\mathcal{L}(u_m, \tilde{\theta}_m) = \prod_{n=1}^T P_{i_{n-1}i_n}(u_m, \tilde{\theta}_m),$$

where  $P_{ij}(u_m, \tilde{\theta}_m)$  denotes the state transition probability associated with  $u_m$  and  $\tilde{\theta}_m$ . To improve the estimate  $\tilde{\theta}_m$ , we make a small stochastic approximation-type adjustment in the direction of update  $\nabla_{\tilde{\theta}} \mathcal{L}(u_m, \tilde{\theta}_m)$ . Although expressed in terms of the continuous-time implementation (cf. Eqns. (12-13)), Algorithm 2 accomplishes this *in principle* by adjusting  $\theta_m$  in the direction

$$\sum_{n=1}^T \frac{\nabla_{\tilde{\theta}} P_{i_{n-1}i_n}(u_m, \tilde{\theta}_m)}{P_{i_{n-1}i_n}(u_m, \tilde{\theta}_m)}. \quad (14)$$

The rationale for this is that

$$\nabla_{\tilde{\theta}} \mathcal{L}(u_m, \tilde{\theta}_m) = \mathcal{L}(u_m, \tilde{\theta}_m) \sum_{n=1}^T \frac{\nabla_{\tilde{\theta}} P_{i_{n-1}i_n}(u_m, \tilde{\theta}_m)}{P_{i_{n-1}i_n}(u_m, \tilde{\theta}_m)},$$

where the right hand side of Eqn. (14) differs only by a scaling factor  $\mathcal{L}(u_m, \tilde{\theta}_m)$ . Our numerical results in Section V illustrate the effectiveness of this procedure.

#### B. Analytical Issues with Algorithm 2

The fact that Algorithm 1 converges with probability one to a critical point stems from the observation in [29] that the local variable  $F$  (at the end of a regenerative cycle) is asymptotically consistent with the direction of the gradient of the objective function. The analyses of [29] and [30] (relating to the  $i^*$ -adaptation procedure) involve a careful accounting of the errors associated with imperfect estimates of average reward of the process  $\lambda(u_m)$ . The analysis of Algorithm 2 is complicated by the fact that new errors are introduced from imperfect estimates of the parameter vector  $\theta^*$ . In order to prove that Algorithm 2 offers any beneficial convergence properties, we must resolve at least two issues:

- 1) *Does the optimization procedure allow enough observation of the process in order to guarantee convergence?*
  - From the theory of adaptive control of Markov chains (see, for example, [35], pages 121-122), conditions are known under which the optimization procedure will converge when the maximum likelihood estimator is used. In Algorithm 2, however, we use a gradient procedure to estimate  $\theta^*$ , and convergence of the estimate remains to be shown. Moreover, if  $\{\tilde{\theta}_m\}_{m=0}^{\infty}$  converges in Algorithm 2, it may converge to a locally optimal estimate.
- 2) *Does the error  $(\theta^* - \tilde{\theta}_m)$  introduce enough noise to prevent recovery of the true direction of the gradient?*
  - If the functions  $\alpha_k(u_k, \tilde{\theta})$ , and  $\beta_k(\tilde{\theta})$  are such that

the assumptions in [35] are met, then it is possible to show that the direction of the price vector update deviates from the gradient by a term that can be bounded by  $M\|\theta^* - \tilde{\theta}_m\|$ , for some constant  $M$ . This may be enough to guarantee that the direction of the gradient will be eventually recovered (of course as long as  $\|\theta^* - \tilde{\theta}_m\| \rightarrow 0$ .)

## V. NUMERICAL EVALUATION

Here, we present numerical examples that illustrate the application of Algorithms 1 and 2. Examples 1 and 2 illustrate (1) the convergence of the model-based algorithm (Algorithm 1) assuming perfect knowledge of the arrival rate functions and holding times and (2) the ability of the adaptive algorithm (Algorithm 2) to identify the same set of prices in the presence of uncertainty about some of the model parameters. Examples 3 and 4 illustrate the ability of the adaptive algorithm to track time-varying parameters of the underlying system. All scenarios tested here correspond to situations where the “many small sources” assumption does not apply.

### A. Example 1: Two Classes of Traffic

This example involves two classes of traffic ( $K = 2$ ) in a system with small capacity. The parameters for the example are compiled in Table I. Note that the capacity of the system is  $C = 10.0$  (Mbps) and, since  $m_1 = 1.0$  (Mbps) and  $m_2 = 5.0$  (Mbps), there can be at most 10 active users in the system. Both service classes have a maximum (price-free) request arrival rate of  $\bar{\alpha}_1 = \bar{\alpha}_2 = 10.0$  (requests/s). On the other hand, class-1 demand for services drops (linearly) to zero at a price of  $\check{u}_1 = 1.0$  priceunits (p.u.), whereas class-2 demand for service cuts off at the much higher value  $\check{u}_2 = 10.0$  (p.u.). Even though class-2 users consume five times the bandwidth of class-1 users and stay in the system for the same length of time ( $1/\beta_1 = 1/\beta_2 = 1.0$  (s) on average), the demand for class-2 service is so much higher than class-1 service for a given price that class 2 traffic offers better earning potential to the service provider. In applying Algorithms 1 and 2, we restrict the range of prices for class-1 service to the interval  $u_1 \in [0, .9]$  (p.u.), and, similarly, we restrict the range of prices for class-2 service to the interval  $u_2 \in [0, 9.0]$  (p.u.). Thus, it is impossible for the optimization algorithm to accidentally “turn off” the process of request arrivals in either class. For the adaptive case, we assume that service rate distribution parameters  $\beta_1$  and  $\beta_2$  are unknown.

TABLE I  
PARAMETERS FOR EXAMPLE 1. ( $C = 10$ )

Parameter	Class 1	Class 2
$\alpha_k(u_k)$	$\bar{\alpha}_1(1 - u_1/\check{u}_1)_+$	$\bar{\alpha}_2(1 - u_2/\check{u}_2)_+$
$\bar{\alpha}_k$	10.0	10.0
$\check{u}_k$	1.0	10.0
$\bar{u}_k$	.9	9.0
$m_k$	1.0	5.0
$\beta_k$	1.0	1.0

The results of this example appear in Figs. 1 and 2. Fig. 1 shows the evolution of prices and estimated average reward

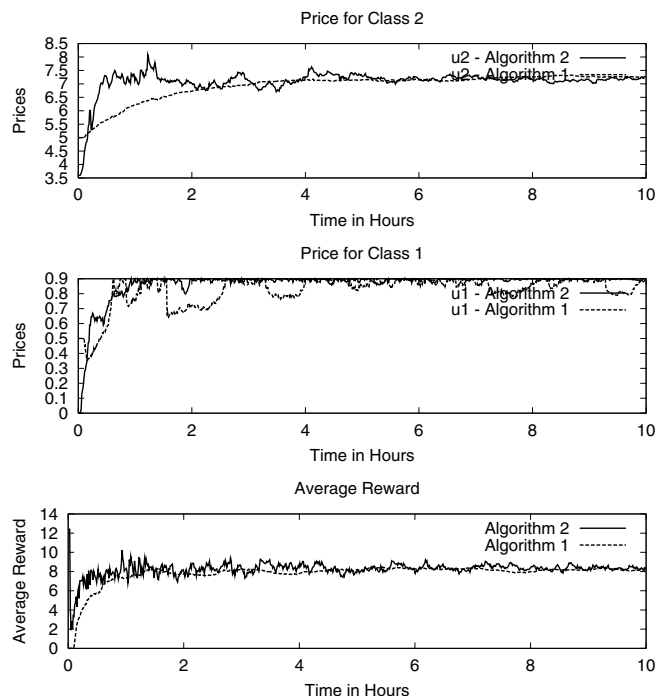


Fig. 1. Evolution of prices and estimated average reward in Example 1.



Fig. 2. Evolution of the estimates of  $\beta$  in Example 1.

over 10 hours for both algorithms. Note that both the model-based and the adaptive algorithms converge to nearly the same set of prices,  $u_1 \approx .9$  and  $u_2 \approx 7.0$ . (This price vector has the effect of shutting off class-1 traffic as much as possible, leaving as much room as possible for the other, more profitable, service class.) Fig. 2 shows the evolution of the estimates of  $\beta_1$  and  $\beta_2$  in Algorithm 2. Note that the estimates eventually converge to the true value  $\beta_1 = \beta_2 = 1.0$  ( $s^{-1}$ ). We point out the relatively slow rate of convergence of both algorithms, especially with regard to price. On the other hand, even though it takes a long time for prices to settle, the average reward of the process seems to lock onto its final value relatively quickly.

### B. Example 2: Three Classes of Traffic

This example involves three classes of traffic, with guaranteed rates  $m_1 = 1$ ,  $m_2 = 10$ ,  $m_3 = 5.0$  (Mbps), in a system with capacity  $C = 20$  (Mbps). The full set of parameters for the example are compiled in Table II. As in Example 1, prices are optimized both for the case where all the parameters are

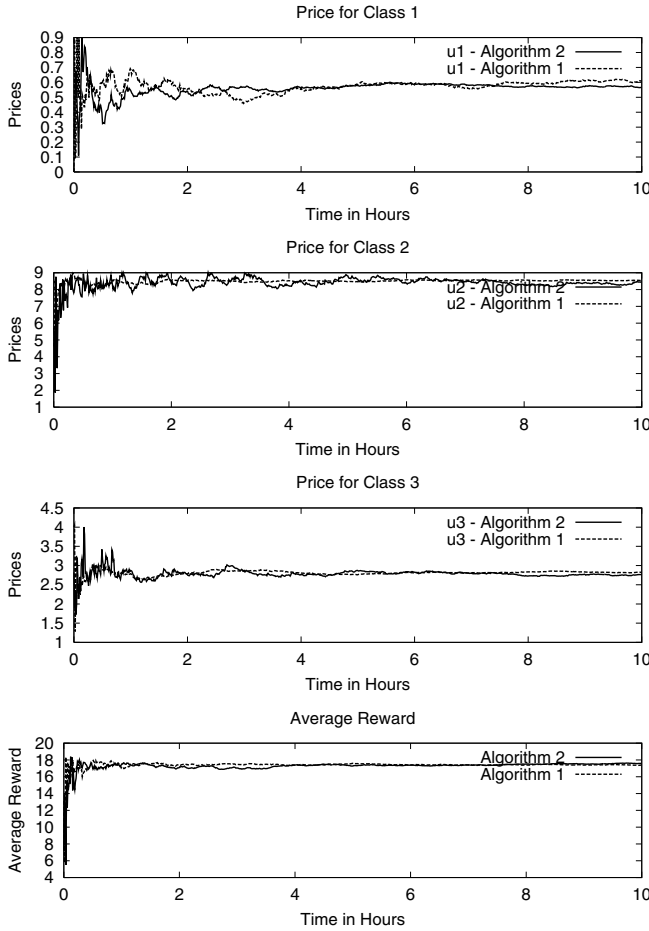


Fig. 3. Evolution of prices and estimated average reward in Example 2.

known (Algorithm 1) and where the service rate parameters  $\beta_1, \beta_2$ , and  $\beta_3$  are unknown (Algorithm 2).

TABLE II  
PARAMETERS FOR EXAMPLE 2. ( $C = 20$ )

Parameter	Class 1	Class 2	Class 3
$\alpha_k(u_k)$	$\bar{\alpha}_1(1 - u_1/\bar{u}_1)_+$	$\bar{\alpha}_2(1 - u_2/\bar{u}_2)_+$	$\bar{\alpha}_3(1 - u_3/\bar{u}_3)_+$
$\bar{\alpha}_k$	10.0	10.0	10.0
$\bar{u}_k$	1.0	10.0	5.0
$\bar{u}_k$	.9	9.0	4.8
$m_k$	1.0	10.0	5.0
$\beta_k$	10.0	1.0	5.0

The results of this example appear in Figs. 3 and 4. Fig. 3 shows the evolution of prices and estimated average reward over 10 hours of operation. Note that, as before, both the model-based and the adaptive algorithms converge to nearly the same set of prices. Fig. 2 shows the evolution of the estimates of  $\beta_1, \beta_2$ , and  $\beta_3$  in Algorithm 2, with all three parameters eventually tracking their actual values.

### C. Example 3: Tracking Abrupt Changes

This example seeks to explore the ability of Algorithm 2 to track abrupt changes in model parameters. As in Example 1, we consider a system with two classes of service and limited

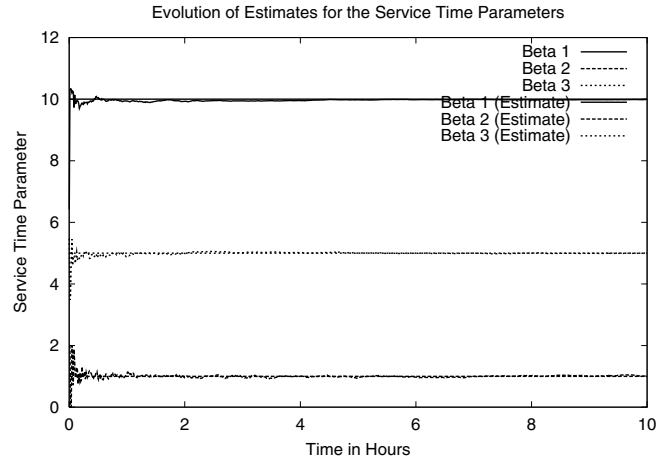


Fig. 4. Evolution of the estimates of  $\beta_1, \beta_2, \beta_3$  in Example 2.

service capacity  $C = 10$  (Mbps). The parameters of the model are expressed in Table III. Note that the service rate parameters  $\beta_1$  and  $\beta_2$  are given as functions of time, i.e. for the first third of the experiment  $\beta_1 = .1$  and  $\beta_2 = 1.0$ , for the second third of the experiment  $\beta_1 = 1.0$  and  $\beta_2 = .1$ , and for the final third of the experiment we return to  $\beta_1 = .1$  and  $\beta_2 = 1.0$ . Note that, throughout the experiment, both service classes have identical demand characteristics (i.e. their arrival rate functions  $\alpha_k(\cdot)$  are the same). Finally, to accommodate the time-varying nature of the problem we modify Algorithm 2 slightly by imposing a constant stepsize rule,  $\gamma_k = \gamma_0$  for some  $\gamma_0$  small.

TABLE III  
PARAMETERS FOR EXAMPLE 3. ( $C = 10$ )

Parameter	Class 1	Class 2
$\alpha_k(u_k)$	$\bar{\alpha}_1(1 - u_1/\bar{u}_1)_+$	$\bar{\alpha}_2(1 - u_2/\bar{u}_2)_+$
$\bar{\alpha}_k$	10.0	10.0
$\bar{u}_k$	1.0	1.0
$\bar{u}_k$	.9	.9
$m_k$	1.0	1.0
$\beta_k$		
$t \in [0, 6)$ hours	.1	1.0
$t \in [6, 12)$ hours	1.0	.1
$t \in [12, 20)$ hours	.1	1.0

The results of this example appear in Figs. 5 and 6. Fig. 5 shows the evolution of prices and estimated average reward over the entire 20 hours of operation. Note that the prices for classes 1 and 2 appear to alternate after the switch in  $\beta_1$  and  $\beta_2$  at  $t = 6$  hours and  $t = 12$  hours. Thus, Algorithm 2 seems to be making the correct response to the instantaneous jump in the service rate parameters. (Note that the estimated average reward dips during the transient after each switch.) Fig. 6 shows the evolution of the estimates of  $\beta_1$  and  $\beta_2$ . Note that with the constant step size rule the estimates of  $\beta_1$  and  $\beta_2$  quickly track the instantaneous jumps in these parameters, although the estimates remain somewhat noisy throughout.

### D. Example 4: Tracking Smooth Changes

In this example we explore the ability of Algorithm 2 to track smooth changes in model parameters. As in Examples



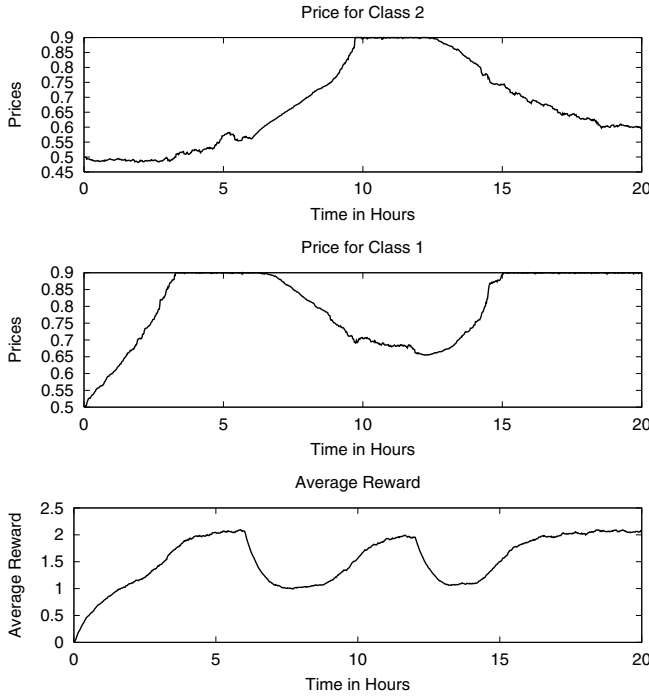


Fig. 5. Evolution of prices and estimated average reward in Example 3.

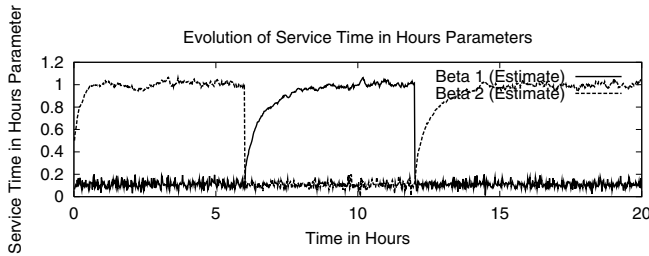


Fig. 6. Evolution of the estimates of  $\beta_1$  and  $\beta_2$  in Example 3.

1 and 3, we consider a system with two service classes and limited capacity  $C = 10$  (*Mbps*). The parameters of the model are expressed in Table IV. As in Example 3, the service rate parameters  $\beta_1$  and  $\beta_2$  are given as functions of time. i.e. for the first third of the experiment  $\beta_1 = .1$  and  $\beta_2 = 1.0$ , for the second third of the experiment  $\beta_1$  increases linearly in time (on average) to  $\beta_1 = 1.0$  and  $\beta_2$  decreases linearly in time (on average) to  $\beta_2 = .1$ , and for the final third of the experiment  $\beta_1$  and  $\beta_2$  hold at 1.0 and  $.1$  ( $s^{-1}$ ), respectively. The demand characteristics for both service classes in this example are time-invariant and are the same as in Example 3.

The results of this example appear in Figs. 7 and 8. Fig. 7 shows the evolution of prices and estimated average reward over the entire 20 hours of operation. Interestingly, from the plot of the average reward estimate, Algorithm 2 appears to be able to extract a higher average reward during the transient, especially where both service classes approach the same (relatively small) expected holding time  $1/\beta_1 = 1/\beta_2 \approx 2$  (*s*). Fig. 8 shows the evolution of the estimates of  $\beta_1$  and  $\beta_2$ . Again, with the constant step size rule, the estimates of  $\beta_1$  and

TABLE IV  
PARAMETERS FOR EXAMPLE 4. ( $C = 10$ ).

Parameter	Class 1	Class 2
$\alpha_k(u_k)$	$\bar{\alpha}_1(1 - u_1/\bar{u}_1)_+$	$\bar{\alpha}_2(1 - u_2/\bar{u}_2)_+$
$\bar{\alpha}_k$	10.0	10.0
$\bar{u}_k$	1.0	1.0
$\tilde{u}_k$	.9	.9
$m_k$	1.0	1.0
$\beta_k$		
$t \in [0, 6)$ hours	.1	1.0
$t \in [6, 12)$ hours	(linear increase)	(linear decrease)
$t \in [12, 20)$ hours	1.0	.1

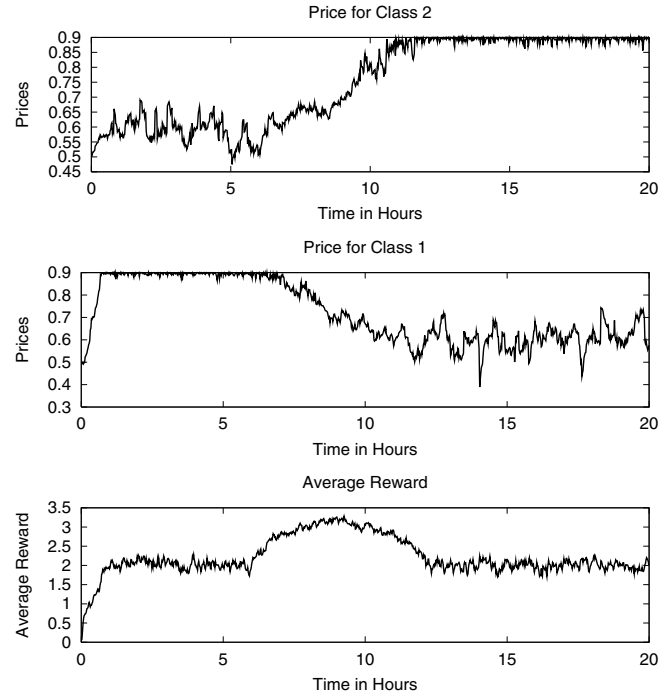


Fig. 7. Evolution of prices and estimated average reward in Example 4.

$\beta_2$  are able to track the smooth changes in these parameters between  $t = 6$  hours and  $t = 12$  hours. As before, however, the constant stepsize rule gives rise to somewhat noisy estimates of the parameters.

## VI. CONCLUSIONS

Seeking to develop efficient and robust pricing mechanisms for network services, we have adapted the simulation-based optimization procedure of [29], [30] for on-line use in continuous time. The most natural implementation of this scheme results in a model-based procedure (Algorithm 1) which is guaranteed to converge to a critical point assuming that all of the arrival and service rates of the underlying Markov process are known. Toward the goal of being robust to parametric uncertainty, we have also developed an adaptive procedure (Algorithm 2) which allows for the simultaneous estimation of model parameters. Both algorithms are efficient in the sense that they do not require that the entire history of the process be stored in order to make adjustments toward an optimal set of prices. In Algorithm 1 the memory requirements are  $O(K)$ ,

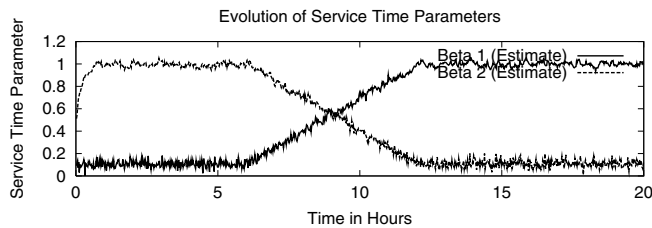


Fig. 8. Evolution of the estimates of  $\beta_1$  and  $\beta_2$  for Example 4.

where  $K$  is the number of service classes, and in Algorithm 2 the memory requirements are  $O(K + q)$ , where  $q$  is the number of uncertain parameters in the model. Our numerical examples illustrate that our on-line algorithms are able to adjust prices toward optimal levels, even in the presence of modeling uncertainty, although the convergence appears to be quite slow.

While the results of this study are encouraging, there are many outstanding questions and issues to resolve. We first acknowledge that our numerical examples only provide a very rough proof-of-concept. It remains to be seen how our algorithms (particularly Algorithm 2) perform under more realistic traffic conditions. We have yet to experiment with general arrival and service time distributions, and we also have not experimented with the social welfare objective discussed in Section II. Another primary objective is to establish analytically the convergence properties of Algorithm 2. This has been being done and its publication is under preparation. Our analysis proceeds by first developing an adaptive version of the discrete-time results in [29], [30]. We show that the estimator is strongly consistent and convergence in probability at a rate that prevents the error introduced by the estimation process from altering the analysis of the algorithm [29].

Second, although we have focused on the single link case here, the network case is interesting. Indeed, it is shown in [28] that the asymptotic optimality of static prices still holds in the network case, and thus the algorithms in [29], [30] can be applied off-line to the network case. Nevertheless, some implementation issues have to be addressed in order to deploy the algorithm in an online fashion. For example, the notion of state of the system, has to be communicated across the different origin destination pairs.

#### ACKNOWLEDGMENT

This work is supported in part by a grant from the National Science Foundation (ECS-9875688 (CAREER)) and by a scholarship from Consejo Nacional de Ciencia y Tecnología (CONACYT-68428).

#### REFERENCES

- [1] A. Odlyzko, "The economics of the Internet: Utility, utilization, pricing, and Quality of Service," AT&T Research Lab, Tech. Rep., 1998.
- [2] R. J. Gibbens and F. P. Kelly, "Resource pricing and the evolution of congestion control," *Automatica*, vol. 35, 1999.
- [3] R. Cocchi, S. Shenker, D. Estrin, and L. Zhang, "Pricing in Computer Networks: Motivation, Formulation, and Example," *IEEE/ACM Transactions on Networking*, vol. 1, no. 6, pp. 614–627, 1993.

- [4] C. Courcoubetis, A. Dimakis, and M. I. Reiman, "Providing Bandwidth Guarantees over a Best-Effort Network: Call-Admission and Pricing," *Proceedings of INFOCOM*, 2001.
- [5] C. Courcoubetis, F. P. Kelly, V. A. Siris, and R. Weber, "A Study of Simple Usage-based Charging Schemes for Broadband Networks," *Telecommunication Systems*, vol. 15, no. 3-4, pp. 323–343, 2000.
- [6] C. Courcoubetis, F. P. Kelly, and R. Weber, "Measurement-Based Usage Charges in Communications Networks," *Operations Research*, vol. 48, no. 4, pp. 535–548, July/August 2000.
- [7] C. Courcoubetis and V. A. Siris, "An Approach to Pricing and Resource Sharing for Available Bit Rate (ABR) Services," Institute of Computer Science (ICS), Tech. Rep. 212, 1997.
- [8] C. Courcoubetis, V. A. Siris, and G. D. Stamoulis, "Integration of Pricing and Flow Control for Available Bit Rate Services in ATM Networks," *Proceedings of IEEE Globecom*, 1996.
- [9] S. H. Low and P. P. Varaiya, "A New Approach to Service Provisioning in ATM Networks," *IEEE/ACM Transactions on Networking*, vol. 1, no. 5, pp. 547–553, October 1993.
- [10] R. Sinha, S. Kalyanaraman, and T. Ravichandran, "Dynamic Capacity Contraction: A Framework for Congestion Sensitive Pricing," *Submitted to ICCS*, 2000.
- [11] J. R. Perkins and R. Srikant, "The Role of Queue Length Information in Congestion Control and Resource Pricing," *Proceedings of the 38th. Conference on Decision and Control*, 1999.
- [12] D. Ferguson, Y. Yemini, and C. Nikolau, "Microeconomic Algorithms for Load Balancing in Distributed Computer Systems," *Proceedings of DCS*, 1988.
- [13] P. B. Key and D. R. McAuley, "Differential QoS and Pricing in Networks: where flow-control meets game theory," *IEE Proceedings on Software*, March 1999.
- [14] N. Semret, R. R. F. Liao, A. T. Campbell, and A. A. Lazar, "Market Pricing of Differentiated Internet Services," Columbia University, Tech. Rep., 1999.
- [15] P. Marbach, "Differentiated Services Networks: Pricing and Software Agents," *International Journal of Parallel and Distributed Systems and Networks; Special Issue on the Application of Intelligent Control and Optimization to Communications Systems and Networks*, 2001.
- [16] —, "Pricing Differentiated Services Networks: Bursty Traffic," *Proceedings of INFOCOM*, 2001.
- [17] —, "The Role of Pricing in Differentiated Services Networks," *submitted*, 2002.
- [18] J. Crowcroft and P. Oechslin, "Differentiated End-to-End Internet Services Using a Weighted Proportional Fair Sharing TCP," *CCR*, vol. 28, no. 3, July 1998.
- [19] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, "TCP Enhancements for an Integrated Services Internet," I.B.M. T.J. Watson Research Center, Tech. Rep. RC 20618, 1996.
- [20] —, "Understanding TCP Dynamics in a Differentiated Services Internet," *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 173–187, April 1999.
- [21] F. Kelly, "Charging and Rate Control for Elastic Traffic," *European Transactions on Telecommunications*, vol. 8, pp. 33–37, 1997.
- [22] R. J. La and V. Anantharam, "Charge-Sensitive TCP and Rate Control in the Internet," *Proceedings of INFOCOM '2000*, 2000.
- [23] X. Lin and N. B. Shroff, "Simplification of Network Dynamics in Large Systems," Purdue University, Tech. Rep., 2001.
- [24] T. Basar and R. Srikant, "Revenue-maximizing pricing and capacity expansion in a many-users regime," *Proceedings of IEEE INFOCOM*, 2002.
- [25] D. Ferrari and L. Delgrossi, "Charging for QoS," *IEEE Transactions on Networking*, 1995.
- [26] N. Keon and G. Anandalingam, "Adaptive Flow Control for VoD Using Price Discounts," *Submitted*, 1999.
- [27] I. C. Paschalidis and J. N. Tsitsiklis, "Congestion-Dependent Pricing of Network Services," *IEEE/ACM Transactions on Networking*, vol. 8, no. 2, pp. 171–184, 2000.
- [28] I. C. Paschalidis and Y. Liu, "Pricing in Multiservice Loss Networks: Static Pricing, Asymptotic Optimality, and Demand Substitution Effects," *IEEE/ACM Transactions on Networking*, vol. 10, no. 3, pp. 425–438, 2002.
- [29] P. Marbach and J. N. Tsitsiklis, "Simulation-Based Optimization of Markov Reward Processes," *IEEE Transactions on Automatic Control*, 2001.

- [30] E. Campos-Nanez and S. D. Patek, "SIE-020003: Dynamically Identifying Regenerative Cycles in Simulation-Based Optimization Algorithms for Markov Chains," Department of Systems and Information Engineering, University of Virginia, Tech. Rep., 2002.
- [31] D. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 1995, vol. II.
- [32] E. Campos-Nanez and S. D. Patek, "On Improving the Performance of Simulation-Based Algorithms for Average Reward Processes with Application to Network Pricing," *Proceedings of the 2001 Winter Simulation Conference*, 2001.
- [33] —, "SIE-020004: On-line Tuning of Prices for Network Services," University of Virginia, Tech. Rep., 2002.
- [34] V. S. Borkar and P. Varaiya, "Adaptive Control of Markov Chains, I: Finite Parameter Set," *IEEE Transactions on Automatic Control*, no. AC-24, pp. 953–957, 1979.
- [35] V. S. Borkar, *Topics in Controlled Markov Chains*. Longman Scientific & Technical, 1991.