Dynamic Pricing for Bandwidth Provisioning

Uday Savagaonkar
School of Electrical and
Computer Engineering
Purdue University
West Lafayette, IN 47907

Robert L. Givan
School of Electrical and
Computer Engineering
Purdue University
West Lafayette, IN 47907

Edwin K. P. Chong
Department of Electrical and
Computer Engineering
Colorado State University
Fort Collins, CO 80523

Abstract

We consider the problem of pricing for bandwidth provisioning over a single link, where users arrive according to a known stochastic *traffic model*. The network administrator controls the resource allocation by setting a price at every epoch, and each user's response to the price is governed by a demand function. We formulate this problem as a partially observable Markov decision process (POMDP), and explore two novel pricing schemes—reactive pricing and spot pricing—and compare their performance to appropriately tuned flat pricing. We use a gradient-ascent approach in all the three pricing schemes. We provide methods for computing the unbiased estimates of the gradient in an online (incremental) fashion. Our simulation results show that our novel schemes take advantage of the known underlying traffic model and significantly outperform the model-free pricing scheme of flat pricing.

I Introduction

Bandwidth trading is becoming increasingly important as many companies want to sell their unused bandwidth. A key problem in completing such trades is to have a good online pricing scheme. Various pricing schemes have been discussed in the literature for scenarios with fixed sets of users [1][2][3]. Pricing schemes appropriate for various problems with dynamic user arrivals have also been developed under various network settings [4][5][6]. But these schemes deal with restrictive traffic models (e.g. Poisson arrivals) and were designed for fixed-bandwidth dialup connections. Our model more naturally allows the users' bandwidth demands to change with price as is typical in bandwidth trading (in previous models, the arrival rate can change in response to price, but each arrival purchases a fixed-bandwidth connection).

We consider the problem of optimal pricing for bandwidth provisioning, in which we restrict our attention to the case where all the resource is controlled by a single broker (typically the network administrator) and is sold to dynamically arriving users on demand. We assume users arrive according to a known stochastic traffic model, and explore methods to exploit the knowledge of this model. The broker controls the price to be charged for the bandwidth. Subject to availability, the amount of bandwidth the users purchase is dictated by their respective demand functions, which we assume are known to the broker. The goal of the broker is to set the price over time so as to maximize its average revenue. We explore two novel pricing schemes—reactive pricing and spot pricing—and compare their performance to appropriately tuned flat pricing. Through simulations we show that the new pricing schemes, which exploit the underlying traffic model, provide significant revenue improvement over the model-free scheme of flat pricing.

We use the following notation. If $\vec{\theta}$ is a vector, θ^i denotes i^{th} component of $\vec{\theta}$. Contrarily, if α is a scalar, α^i denotes i^{th} power of α. Also, for brevity, we present the proofs of the various results elsewhere [7].

II Problem Formulation

A Problem description

We consider a dynamic market in which the resource being traded is the bandwidth over a single link. We assume that the maximum bandwidth available on the link is B. Even though we focus on the single-link scenario, the algorithms we present can be extended to multiple-link, end-to-end trades. We consider a traffic model in which the user arrivals and departures are driven by a discrete-time Markov process $\boldsymbol{S}(\cdot)$, called the *traffic-state process.* The finite state space, $\mathbb{S}$, of this process is made up of elements called *traffic states.* We assume that for each state $s \in \mathbb{S}$, the number of calls arriving in any epoch is a Poisson random variable with mean λ_s. We also assume that for each call arriving in state $s \in \mathbb{S}$, the call-holding time is a geometric random variable with mean α_s. The call-holding time of a call is declared as soon as the call arrives. Note that the Poisson or geometric assumptions are not critical, and in fact can be replaced by any distribution that depends only on the current traffic state.

We characterize a call i, $i \in \mathbb{Z}_+$, by a pair of random variables, $\langle \boldsymbol{a}_i, \boldsymbol{d}_i \rangle$, where $\boldsymbol{a}_i$ represents the (integral) time of arrival of call i and $\boldsymbol{d}_i$ represents the (integral) duration of call i. For call i, we define a discrete-time stochastic

This research is supported in part by DARPA/ITO under contract F30602-00-2-0552 and by NSF grants 9977981-IIS, 0093100-IIS, 0098089-ECS, and 0099137-ANI. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of DARPA, NSF, or the U.S. Government.

process $\boldsymbol{A}_i(\cdot)$ as follows:

$$\boldsymbol{A}_i(k) = \begin{cases} 1 & \text{if } \boldsymbol{a}_i \leq k < \boldsymbol{a}_i + \boldsymbol{d}_i \\ 0 & \text{otherwise} \end{cases} . \qquad (1)$$

In other words, $\boldsymbol{A}_i$ is one over the duration of the call, and zero everywhere else.

When user i arrives, bandwidth allocation is performed as follows. The network administrator observes the current *system state* (to be defined formally later), and declares the price per unit time per unit bandwidth to the user. All users are assumed to purchase bandwidth according to a single *demand function* $D(\cdot)$ describing the amount of resource purchased at each price[1]. We assume that the demand function is compactly-supported, strictly-decreasing, and differentiable over the interval of its support. The demand function determines how much bandwidth the user would ideally purchase at the current price, and its derivative is used by our algorithms in learning locally optimal policies by gradient-ascent tuning. The network administrator next checks if there is enough resource available to satisfy the demand of each of the newly arrived users at the current price. If that is the case, each of the new users gets the requested amount of resource and pays a cost according to the current price. Otherwise, the network administrator divides the available bandwidth equally amongst all the users that have arrived in the current decision epoch, and charges them for the bandwidth it has sold. Bandwidth once sold to a user cannot be reclaimed before the user leaves the system, and the initial allocation to a new user is consumed at every time epoch by that user for the duration of the call. We assume that a user willing to purchase a given bandwidth at a given price is also willing to purchase any smaller amount at the same unit price–there is no minimum-bandwidth requirement in our model.

As described above, when combined with information about the available resource and the number of new users in the system, the demand function $D(\cdot)$ uniquely tells us the amount of resource the user receives. We call this amount the *effective demand function* of the user, and denote it by $\mathcal{D}(p, r, n)$, where $p \in \mathbb{R}_+$ is the price, $r \in \mathbb{R}_+$ is the available resource, and $n \in \mathbb{Z}_+$ is the number of users arriving in the current decision epoch.

Given the resource-allocation mechanism as described above, the aim of the network administrator is to set the link prices $p(\cdot)$ (a function of time) so that the expected revenue is maximized. In other words, the network administrator should solve the following optimization problem:

$$\max \quad \lim_{H \to \infty} E\left[\frac{1}{H} \sum_{k=0}^{H-1} \sum_{i \in \mathbb{Z}_+} (\boldsymbol{A}_i(k) b_i p(\boldsymbol{a}_i))\right], \qquad (2)$$

where $p(\cdot)$ are the decision variables, and b_i is the bandwidth allocated to user i.

[1]We assume this single demand function for simplicity here, but this assumption can be relaxed by treating each user's demand function as a random variable whose value is declared at arrival.

To tackle such problems, a heuristic, called the *rolling-horizon* approach, is widely used [8]. In this approach, instead of considering the steady-state expected reward as the objective function, a finite horizon length H is fixed. At decision epoch t, an action is chosen to maximize the expected reward over the horizon from t to $t + H - 1$. In this framework the objective of the network administrator at decision epoch t is:

$$\max \quad E\left[\sum_{k=t}^{t+H-1} \sum_{i \in \mathbb{Z}_+} (\boldsymbol{A}_i(k) b_i p(\boldsymbol{a}_i))\right], \qquad (3)$$

The pricing schemes that we introduce in Section III use this framework for computing the optimal prices.

B Partially Observable Markov Decision Process (POMDP) formulation

State Space: A state $x \in X$ is a triple $\langle s_x, n_x, o_x \rangle$, where $s_x \in S$ gives the arrival process state, and n_x and o_x are multisets of active user descriptions, as follows. An active user description is a pair $\langle r, b \rangle$ of a natural number $r > 0$ and a real number b in $[0, B]$. The user description $\langle r, b \rangle$ represents a user consuming resource b per time epoch for remaining duration of activity r. The set n_x represents the users that have just arrived, and we require that each member $\langle r, b \rangle$ of n_x has $b = 0$ as no bandwidth has been purchased for these users yet. The set o_x represents the users still in the system from previous arrivals.

Action Space: At any decision epoch, the network administrator is allowed to control the price for the link bandwidth. Thus, our action space $\mathbb{A}$ is just $\mathbb{R}_+$.

Transition Law: Using the statistics of the traffic-state process and the various assumptions made in Section II-A, one can easily write down the transition law (see [7]).

Reward Structure: Let $x = \langle s_x, n_x, o_x \rangle$ be the current state of the system and u be the action chosen. Let $\mathcal{L}(x)$ be the available resource at state x, $B - \sum_{\langle r,b \rangle \in o_x} b$, and $\mathcal{N}(x)$ be the number of arrivals at state x, $|n_x|$. Then, the one-step reward $g : \mathbb{X} \times \mathbb{A} \to \mathbb{R}$ is given by

$$g(x, u) = \sum_{\langle r,b \rangle \in n_x} D(u, \mathcal{L}(x), \mathcal{N}(x)) ur. \qquad (4)$$

Using this definition of one-step reward, our objective is to find the policy that maximizes the expected average reward.

Observation Space: We treat the traffic-state part s_x of the state x as unobservable. Specifically, the observation space $\mathbb{O}$ is the set of pairs $o = \langle n_o, o_o \rangle$ of multisets of user descriptions, corresponding to the observation of the state components n_x and o_x.

Observation Kernel: The observation kernel gives, for each state x and action u, a probability distribution over the observation space $\mathbb{O}$ that assigns probability one to the observation $\langle n_x, o_x \rangle$.

C Completely-observable MDP (COMDP) formulation

A POMDP can be converted to an equivalent COMDP whose state space consists of probability distributions over the state space of the POMDP. In the case of our problem, the only un-observable part of the POMDP state x is the traffic state of the system, s_x. Thus, our POMDP can be converted to a COMDP whose state $\tilde{x}$ is a three tuple $\langle \vec{I}_{\tilde{x}}, n_{\tilde{x}}, o_{\tilde{x}} \rangle$, where $n_{\tilde{x}}$ and $o_{\tilde{x}}$ are as explained in the definition of the partially observable state space $\mathbb{X}$. The component $\vec{I}_{\tilde{x}}$ is an $|\mathbb{S}|$-dimensional vector representing a probability distribution over $\mathbb{S}$. Thus, $I^s_{\tilde{x}}$ indicates the probability of being in traffic state s. The state $\tilde{x}$ of this COMDP is also called the *belief state* of the system. As in the previous section, we also extend the functions $\mathcal{L}(\tilde{x})$ and $\mathcal{N}(\tilde{x})$ to represent the leftover bandwidth and number of arrivals at belief state $\tilde{x}$. These are not random because the relevant state components are fully observed. The reward function $g(\tilde{x}, u)$ can also be extended in a similar fashion [7].

III Pricing Schemes

A Flat pricing

In this naive scheme, the network administrator charges the same price at all decision epochs. Though closed-form formulas are available to compute the optimal flat price for simple traffic models [4][6], no such formulas exist for general traffic models. Here we present a stochastic gradient-ascent scheme to solve the problem in (3). Without loss of generality[2], we restrict our attention to decision epoch $t = 0$. To facilitate the discussion, we define the set $\mathfrak{A}[t_1, t_2]$ of users that are active for at least one epoch between the epochs t_1 through t_2. Thus,

$$\mathfrak{A}[t_1, t_2] \triangleq \{i : \boldsymbol{A}_i(k) = 1, \text{ for some } k \in [t_1, t_2]\}. \quad (5)$$

Also, associated with call $i \in \mathfrak{A}[0, H-1]$, we define a random variable $\tilde{\boldsymbol{d}}_i$ to be the duration of that part of call i that overlaps with the interval 0 to $H-1$, given by $\min\{\boldsymbol{d}_i, \boldsymbol{d}_i + \boldsymbol{a}_i, H - \boldsymbol{a}_i, H\}$. In addition, for notational convenience, we extend the effective demand function $\mathcal{D}(p, r, n)$ to a user-specific effective demand function that is specialized to the conditions for user i, written $\mathcal{D}(p, i)$ and defined by $\mathcal{D}(p, \mathcal{L}(\tilde{\boldsymbol{X}}_{a_i}), \mathcal{N}(\tilde{\boldsymbol{X}}_{a_i}))$, where $\tilde{\boldsymbol{X}}_k$ is the system belief state at time k.

Now in this notation, setting $t = 0$ in (3), and noting that in flat pricing the price is constant, say p, the objective function for this problem can be written as the expectation of the following stochastic function:

$$\boldsymbol{V}_H^{flat}(p) = \sum_{i \in \mathfrak{A}[0, H-1]} \mathcal{D}(p, i) \times p \times \tilde{\boldsymbol{d}}_i \ , \quad (6)$$

where H is the finite horizon.

[2]We note that times may be negative in our formulation.

Theorem 1 *The stochastic function $\boldsymbol{V}_H^{flat} : \mathbb{R}_+ \to \mathbb{R}$ is differentiable a.e. on $\mathbb{R}_+$ with probability one. Moreover, $\partial \boldsymbol{V}_H^{flat}(p)/\partial p$ gives an unbiased estimate of $\partial E[\boldsymbol{V}_H^{flat}(p)]/\partial p$.*

Theorem 1 implies that one can use the derivative of $\boldsymbol{V}_H^{flat}(\cdot)$ along an observed sample path of duration H as an estimate of the gradient of the objective function. Thus, to maximize the objective function online, we repeat the process of gradient estimation many times, and every time we estimate the gradient, we take a step in the direction of the gradient. A step size of η_k such that $\sum \eta_k = \infty$ and $\sum \eta_k^2 < \infty$ is known to be appropriate for such algorithms [9].

The derivative of $\boldsymbol{V}_H^{flat}(p)$ along an observed sample path can be computed in an incremental fashion as follows. Define $\boldsymbol{v}_L(\cdot)$, for $0 \le L < H$, recursively as follows:

$$\begin{aligned} \boldsymbol{v}_0(p) &= \sum_{\{i \in \mathfrak{A}[0,0]\}} \left(\mathcal{D}(p,i) \times p \times \tilde{\boldsymbol{d}}_i \right) \\ \boldsymbol{v}_L(p) &= \sum_{\{i : \boldsymbol{a}_i = L\}} \left(\mathcal{D}(p,i) \times p \times \tilde{\boldsymbol{d}}_i \right) + v_{L-1}(p) \end{aligned} \quad (7)$$

Using this definition, we have $\boldsymbol{V}_H^{flat}(p) = \boldsymbol{v}_{H-1}(p)$, and thus $\boldsymbol{V}_H^{flat}(\cdot)$ can be computed in an incremental fashion as we observe the sample path. Now, differentiating (7), we get,

$$\begin{aligned} \frac{\partial \boldsymbol{v}_L(p)}{\partial p} &= \frac{\partial \boldsymbol{v}_{L-1}(p)}{\partial p} \\ &\quad + \sum_{\{i : \boldsymbol{a}_i = L\}} \left(\mathcal{D}(p,i) + p \times \frac{\partial \mathcal{D}(p,i)}{\partial p} \right) \times \tilde{\boldsymbol{d}}_i . \end{aligned} \quad (8)$$

But from the definition of $\mathcal{D}(p,i)$ it can easily be seen that

$$\frac{\partial \mathcal{D}(p,i)}{\partial p} = \begin{cases} \frac{\partial D(p)}{\partial p} & \text{if not congested,} \\ \frac{1}{\mathcal{N}(\tilde{\boldsymbol{X}}_{\boldsymbol{a}_i})} \frac{\partial \mathcal{L}(\tilde{\boldsymbol{X}}_{\boldsymbol{a}_i})}{\partial p} & \text{otherwise,} \end{cases} \quad (9)$$

where call i is considered "congested" if and only if $\sum_{\{j : \boldsymbol{a}_j = \boldsymbol{a}_i\}} D(p) > \mathcal{L}(\tilde{\boldsymbol{X}}_{\boldsymbol{a}_i})$.

The available bandwidth $\mathcal{L}(\boldsymbol{X}_L)$ at time L, in turn evolves according to the following discrete-time equation,

$$\mathcal{L}(\tilde{\boldsymbol{X}}_k) = \mathcal{L}(\tilde{\boldsymbol{X}}_{k-1}) - \sum_{\{i : \boldsymbol{a}_i = k-1\}} \mathcal{D}(p,i) + \sum_{\{i : \boldsymbol{a}_i + \boldsymbol{d}_i = k\}} \mathcal{D}(p,i), \quad (10)$$

which on differentiation gives,

$$\frac{\partial \mathcal{L}(\tilde{\boldsymbol{X}}_k)}{\partial p} = \frac{\partial \mathcal{L}(\tilde{\boldsymbol{X}}_{k-1})}{\partial p} - \sum_{\{i : \boldsymbol{a}_i = k-1\}} \frac{\partial \mathcal{D}(p,i)}{\partial p} + \sum_{\{i : \boldsymbol{a}_i + \boldsymbol{d}_i = k\}} \frac{\partial \mathcal{D}(p,i)}{\partial p}. \quad (11)$$

Equations (9) and (11) are discrete-time causal equations, that can be implemented in an online fashion to compute the respective derivatives, which when combined with (8), can be used to compute $\partial \boldsymbol{V}_H^{flat}(p)/\partial p$ online.

B Reactive Pricing

In our remaining two pricing approaches, the price being charged can vary from epoch to epoch at the administrator's discretion. The administrator decides the appropriate price for a particular epoch using the underlying *known* traffic model. In reactive pricing, the network administrator associates a price with each underlying traffic state, i.e., maintains a vector $\vec{\theta}$ of $|\mathbb{S}|$ prices. Because the arrival process is not fully observable, at each epoch, the administrator chooses a *state estimate* $\tilde{s}$ according to the probability distribution $\vec{I}_{\tilde{x}}$—this defines the stochastic process $\tilde{\boldsymbol{S}}(\cdot)$ giving $\tilde{s}$ over time. The administrator then chooses the component $\theta^{\tilde{s}}$ of $\vec{\theta}$ corresponding to $\tilde{s}$ as the current price.

Thus, we can view the vector $\vec{\theta}$ as a design parameter to be tuned by gradient ascent. Define

$$\boldsymbol{V}_H^{reactive}(\vec{\theta}) = \sum_{i \in \mathfrak{A}[0,H-1]} \left(\mathcal{D}(\theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)}, i) \times \theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)} \times \tilde{\boldsymbol{d}}_i \right).$$

Our goal is to find the $\vec{\theta}$ that maximizes $E[\boldsymbol{V}_H^{reactive}(\cdot)]$. We have the following result:

Theorem 2 *The function $\boldsymbol{V}_H^{reactive} : \mathbb{R}_+^{|\mathbb{S}|} \to \mathbb{R}$ is differentiable almost everywhere on $\mathbb{R}_+^{|\mathbb{S}|}$ with probability one. Also, $\partial \boldsymbol{V}_H^{reactive}(\vec{\theta})/\partial\theta^s$ is an unbiased estimate of $\partial E[\boldsymbol{V}_H^{reactive}(\vec{\theta})]/\partial\theta^s$ for all $s \in \mathbb{S}$.*

Theorem 2 is an analog of Theorem 1 for the reactive-pricing scheme. Using this theorem, an on-line, incremental gradient-based algorithm by generalizing the method given in the previous section to tune each component of $\vec{\theta}$ in place of tuning the scalar p. This generalization is fairly straightforward, and is given by the following variants of equations 7 to 11.

$$\begin{aligned} \boldsymbol{v}_0(\vec{\theta}) &= \sum_{\{i \in \mathfrak{A}[0,0]\}} \left(\mathcal{D}(\theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)}, i) \times \theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)} \times \tilde{\boldsymbol{d}}_i \right) \\ \boldsymbol{v}_L(\vec{\theta}) &= \sum_{\{i:\boldsymbol{a}_i = L\}} \left(\mathcal{D}(\theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)}, i) \times \theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)} \times \tilde{\boldsymbol{d}}_i \right) + v_{L-1}(p) \end{aligned} \tag{12}$$

$$\begin{aligned} \frac{\partial \boldsymbol{v}_L(\vec{\theta})}{\partial\theta^s} &= \frac{\partial \boldsymbol{v}_{L-1}(\vec{\theta})}{\partial\theta^s} \\ &+ \sum_{\{i:\boldsymbol{a}_i = L\}} \left(\mathcal{D}(\theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)}, i) \frac{\partial \theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)}}{\partial\theta^s} + \theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)} \frac{\partial \mathcal{D}(\theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)}, i)}{\partial\theta^s} \right) \tilde{\boldsymbol{d}}_i. \end{aligned} \tag{13}$$

$$\frac{\partial \mathcal{D}(\theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)}, i)}{\partial\theta^s} = \begin{cases} 0 & \text{if not congested and } \tilde{\boldsymbol{S}}(\boldsymbol{a}_i) \neq s, \\ \frac{\partial D(\theta^s)}{\partial\theta^s} & \text{if not congested and } \tilde{\boldsymbol{S}}(\boldsymbol{a}_i) = s, \\ \frac{1}{\mathcal{N}(\tilde{\boldsymbol{X}}_{\boldsymbol{a}_i})} \frac{\partial \mathcal{L}(\tilde{\boldsymbol{X}}_{\boldsymbol{a}_i})}{\partial\theta^s} & \text{otherwise,} \end{cases} \tag{14}$$

$$\mathcal{L}(\tilde{\boldsymbol{X}}_k) = \mathcal{L}(\tilde{\boldsymbol{X}}_{k-1}) - \sum_{\{i:\boldsymbol{a}_i = k-1\}} \mathcal{D}(\theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)}, i) + \sum_{\{i:\boldsymbol{a}_i + \boldsymbol{d}_i = k\}} \mathcal{D}(\theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)}, i), \tag{15}$$

$$\begin{aligned} \frac{\partial \mathcal{L}(\tilde{\boldsymbol{X}}_k)}{\partial\theta^s} &= \frac{\partial \mathcal{L}(\tilde{\boldsymbol{X}}_{k-1})}{\partial\theta^s} - \sum_{\{i:\boldsymbol{a}_i = k-1\}} \frac{\partial \mathcal{D}(\theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)}, i)}{\partial\theta^s} \\ &+ \sum_{\{i:\boldsymbol{a}_i + \boldsymbol{d}_i = k\}} \frac{\partial \mathcal{D}(\theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)}, i)}{\partial\theta^s}. \end{aligned} \tag{16}$$

Note that, in this scheme the network administrator keeps track of the belief state to select a state estimate in order to charge an appropriate price at each decision epoch. To do this, the administrator relies on the underlying traffic model. Thus this scheme exploits the traffic model for pricing the bandwidth efficiently.

C Spot Pricing

In a spot-pricing scheme, as in reactive pricing, the network administrator may change the bandwidth price every decision epoch. But here, the administrator is not bound to select a price as a deterministic function of the state estimate $\tilde{s}$. Solving the pricing problem as a POMDP as described in Section II-B results in an optimal spot-pricing policy. In Section II-C, we converted the pricing POMDP to a belief-state MDP. But this belief-state MDP has an uncountable state space, and thus techniques such as value iteration or linear programming (see [10]) cannot be applied in this case.

There are a number of heuristic techniques for solving such MDPs [11][12][13][14]. Here we explore the policy-rollout technique developed by Bertsekas and Castanon [13] because of the simplicity of implementation and availability of an obvious "base policy" for rollout. Here we describe this technique briefly. See [7] and [13] for more details. A policy $\pi = \{\mu_0^\pi, \mu_1^\pi, \ldots\}$ is a sequence of maps $\mu_k^\pi : \tilde{\mathbb{X}} \to \mathbb{A}$, which map a belief state $\tilde{x}$ to action $u = \mu_k^\pi(\tilde{x})$ at time k. Given a policy π, let $\tilde{\boldsymbol{X}}_k$ denote the belief-state trajectory that results from following policy π from start state $\tilde{x}$ and then define

$$\boldsymbol{q}_H^\pi(\tilde{x}, u) = g(\tilde{x}, u) + \sum_{k=1}^{H-1} g(\tilde{\boldsymbol{X}}_k, \mu_k^\pi(\tilde{\boldsymbol{X}}_k)),$$

Also define $Q_H^\pi(\tilde{x}, u) \triangleq E\left[\boldsymbol{q}_H^\pi(\tilde{x}, u) | \tilde{\boldsymbol{X}}_0 = \tilde{x}\right]$. In the policy rollout technique, a "reasonably good" base policy π_0 is chosen. Then based on the current state $\tilde{x}$, an action $u^* = \arg\max_u Q_H^{\pi_0}(\tilde{x}, u)$ is chosen as the current action. Under certain general conditions, such a policy is an improvement over π_0. The following result helps us estimate the derivative of $Q_H^{\pi_0}(\cdot, \cdot)$ with respect to u.

Theorem 3 *The stochastic function $\boldsymbol{q}_H^{\pi_0}(\tilde{x}, u)$ is differentiable with respect to u a.e. on $\mathbb{R}_+$ with probability*

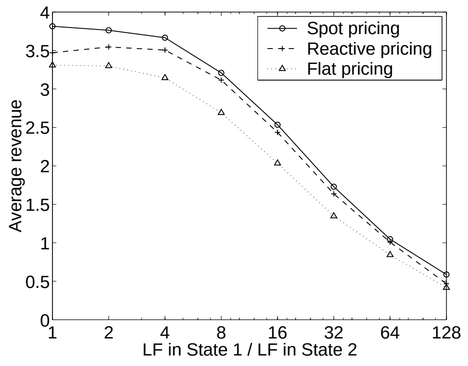

Figure 1: Comparison of the three pricing schemes as a function of the ratio of the load factors.

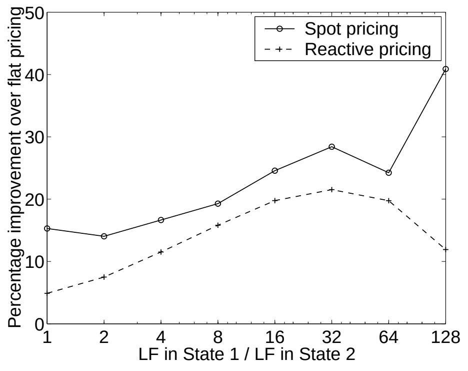

Figure 2: Percentage improvement over the flat-pricing scheme as a function of the ratio of the load factors.

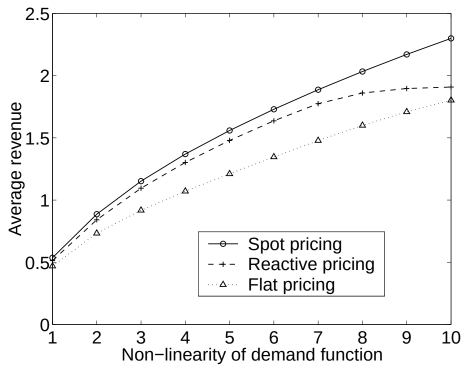

Figure 3: Comparison of the three pricing schemes as a function of the non-linearity of the demand function.

one. Moreover, $\partial \boldsymbol{q}_H^{\pi_0}(\tilde{x}, u)/\partial u$ *gives a conditionally unbiased estimate of* $\partial Q_H^{\pi_0}(\tilde{x}, u)/\partial u$*, conditioned on the event* $\tilde{\boldsymbol{X}}_0 = \tilde{x}$*.*

Thus, to find the current price, we estimate the gradient of $Q_H^{\pi_0}$ by drawing multiple samples with $\tilde{\boldsymbol{X}}_0 = \tilde{x}$ and computing the derivative of $\boldsymbol{q}_H^{\pi_0}(\cdot, \cdot)$ for each sample using a technique similar to the one demonstrated in Section III-A. We then take a step in that direction and repeat this procedure several times until some stopping criterion is met.

Note that the process of drawing samples distinguishes spot pricing from the previous two schemes. Both in flat pricing and reactive pricing, we rely on the past trajectory of the system state to infer the gradient information. In spot pricing, on the other hand, we simulate the *future* of the system by drawing multiple future trajectories using the system belief state and the underlying traffic model for this purpose. As a consequence, the spot-pricing scheme relies heavily on the knowledge of the traffic model.

IV Empirical Results

In this section, we summarize the empirical results briefly. We compare the performance of the three pricing schemes via simulation. Mainly we wish to evaluate the pricing schemes on the basis of (i) how they cope with varying traffic load, and (ii) how they cope with non-linearity in the demand function.

We consider a bandwidth market in which only one vendor is present. The only resource it has for sale is the bandwidth on a single link. We restrict our attention to the case where all the users have a single demand function of the form $-\log(p/k)/k$. We call the parameter k the *non-linearity* of the demand function, as a function with larger value of k is more "non-linear." The users arrive according to a discrete-time Markov Modulated Poisson Process (MMPP) in which the underlying traffic-state process has four states. States 1 and 3 have high traffic load, whereas States 2 and 4 have low traffic load. Formally, we define the load factor in traffic state s as the mean call holding time α_s multiplied by the average number of arrivals λ_s in that state. To evaluate the performance of the three traffic schemes under varying load conditions, we vary the ratio of the load factors in States 1 and 2 (by varying the load factor in State 2). Fig. 1 and Fig. 2 compare the performance of the three traffic schemes as the ratio of the traffic load in States 1 and 2 is changed. It can be seen that spot pricing always performs the best, while flat pricing always performs the worst. Similar results are obtained for Fig. 3 and Fig. 4 where we compare the performance of the three pricing schemes as the non-linearity of the demand function is changed. Also, it can be seen that unlike reactive pricing, spot pricing maintains its advantage even at high non-linearity values. Further evaluation of these techniques is presented in [7].

V Summary

We presented the problem of pricing in a broker-mediated market, where the entire resource is controlled by the broker. We presented two novel pricing schemes: reactive pricing and spot pricing, and compared their performance with that of the flat-pricing scheme. Flat pricing

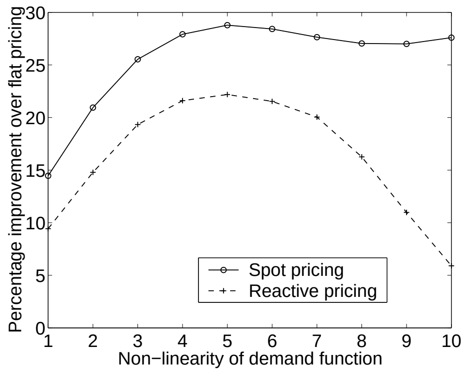

Figure 4: Percentage improvement over the flat-pricing scheme as a function of non-linearity of the demand function.

uses the least system-state information, while the spot-pricing scheme uses the most system-state information. We established that under various traffic conditions and demand structures, the spot-pricing scheme outperforms the reactive-pricing scheme, which in turn outperforms the flat-pricing scheme. The spot-pricing scheme was also shown to take advantage of various conditions, such as varying load and non-linearity of demand function. Even though we focussed on the single-link model, the pricing schemes presented here can immediately be extended to multi-link, end-to-end trades by associating a different price with each link.

References

[1] R. J. Gibbens and F. P. Kelly, "Resource pricing and the evolution of congestion control," *Automatica*, vol. 35, 1995.

[2] S. H. Low and D. E. Lapsley, "Optimization flow control, I: Basic algorithm and convergence," *IEEE/ACM Transactions on Networking*, vol. 7, pp. 861–874, December 1999.

[3] P. Marbach, "Pricing priority classes in a differentiated services network," in *Allerton Conference*, 1999.

[4] 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.

[5] S. D. Patek and E. Campos-Náñez, "Pricing of dialup services: an example of congestion-dependent pricing in the internet," in *Proceedings of IEEE Conference on Decision and Control*, 2000.

[6] Q. Wang and J. M. Peha, "State-dependent pricing and its economic implications," in *Proc. of 7th International Conference on Telecommunication Systems Modeling and Analysis*, (Nashville, Tennessee), pp. 61–71, March 1999.

[7] U. Savagaonkar, E. K. P. Chong, and R. L. Givan, "Optimal pricing for broker-mediated bandwidth market," tech. rep., Purdue University, 2001. Available from http://min.ecn.purdue.edu/∼savagaon/OFFICIAL/my_papers/.

[8] O. Hernández-Lerma and J. B. Lasserre, "Error bounds for rolling-horizon policies in general markov control processes," *IEEE Transactions on Automatic Control*, vol. 35, no. 10, pp. 1118–1124, 1990.

[9] Y.-C. Ho and X.-R. Cao, *Perturbation Analysis of Discrete Event Dynamic Systems.* Kluwer Academic Publishers, 1991.

[10] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming.* New York: John Wiley & Sons, Inc. 1994.

[11] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-dynamic Programming.* Belmont, MA 02178: Athena Scientific, 1996.

[12] P. Marbach and J. N. Tsitsiklis, "Simulation-based optimization of markov reward processes," *IEEE Transactions on Automatic Control*, vol. 46, pp. 191–209, February 2001.

[13] D. P. Bertsekas and D. A. Castanon, "Rollout algorithms for stochastic scheduling problems," *Journal of Heuristics*, vol. 5, pp. 89–108, 1999.

[14] E. K. P. Chong, R. L. Givan, and H.-S. Chang, "A framework for simulation-based network control via hindsight optimization," *Proceedings of 39th IEEE Conference on Decision and Control*, 2000.