Online Pricing for Bandwidth Provisioning in Multi-class Networks

Uday Savagaonkar [a], Edwin K. P. Chong [b,*], Robert L. Givan [a]

[a] *School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907.*

[b] *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.

Key words: pricing, resource allocation, Markov decision processes, stochastic optimization, infinitesimal perturbation analysis.

* Corresponding author, Tel: 970-491-7858, Fax: 970-491-2249
Email addresses: savagaon@ecn.purdue.edu (Uday Savagaonkar), echong@engr.colostate.edu (Edwin K. P. Chong), givan@ecn.purdue.edu (Robert L. Givan).

[1] 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.

[2] A preliminary version of parts of this paper was presented at the 36th Conference on Information Sciences and Systems, Princeton, NJ, 2002.

1 Introduction

Bandwidth trading is becoming increasingly important as many companies want to sell their unused bandwidth. Various companies are setting up electronic agents that could broker such deals. A key problem in completing such trades is to have a good online pricing scheme. A pricing scheme affects the network owner's revenue in two ways. First, it decides how much revenue the owner is going to earn by selling a unit of bandwidth. Second, it affects the amount of bandwidth the users are willing to purchase from the network administrator. Typically, the amount of bandwidth the users purchase from the network administrator is sensitive to the price being charged. Thus, setting up a price for the bandwidth can also be looked at as a way of allocating the bandwidth to various users. In other words, bandwidth trading is closely related to resource allocation.

The latter feature of pricing makes it an effective tool for network control, and it has been used in a wide range of network applications in the literature. Congestion control and resource allocation are closely related to each other because both of these problems deal with distributing the scarce resources amongst the users. Gibbens and Kelly [9] have used pricing for congestion control in a network in which the user actions are governed by utility functions. Marbach [16] has used pricing for congestion control in a differentiated services network. Low and Varaiya [14] have proposed a pricing-based approach to service provisioning in ATM networks. Yaiche et al. [24] have analyzed the problem of pricing broadband networks in a game-theoretic setting. Cao and Shen [4] have shown that cooperative games give better results than leader-follower games when applied to Internet pricing. Thomas et al. [22] have proposed an auction-based framework for service provisioning and have used Scarf's algorithm [21] for determining the optimal prices. Even though these pricing mechanisms show promise, they have been developed to deal with static scenarios. Hence these mechanisms are better suited either for pricing perishable resources or for allocating resources in a preemptive-services network. Also, these mechanisms do not take into account any stochastic knowledge of the system, which the network administrator might have in advance.

Pricing schemes appropriate for various problems with dynamic user arrivals have also been developed under various network settings. Paschalidis and Tsitsiklis [18] have considered the problem of pricing a single link for dial-up connections in a dynamic network setting. They use a dynamic-programming approach for finding the optimal prices, and conclude that under extreme conditions (very large load or very light load) static pricing (where the price does not change with time) can be made to perform almost optimally. Patek and Campos-Náñez [19] have used a similar model and have provided experimental results to show that dynamic pricing (where the price adapts to the system

conditions) cannot give significant advantage over static pricing for large systems. Wang and Peha [23] have used a similar system model and have shown that under certain system assumptions, dynamic pricing can outperform static pricing. But all these schemes deal with restrictive traffic models (e.g., Poisson arrivals) and were designed primarily 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 could change in response to price changes, but each arrival would purchase a fixed-bandwidth connection).

We consider the problem of dynamic 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. As already mentioned, this problem is closely related to that of resource allocation. We assume that the 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 throughout this paper. If $\vec{\theta}$ is a vector, then θ^i denotes i^{th} component of $\vec{\theta}$. On the other hand, if α is a scalar, then α^i denotes i^{th} power of α. We use bold-face notation to denote a random variable. Also, for any set, say $\mathbb{Q}$, we use the notation $|\mathbb{Q}|$ to denote the cardinality of that set.

2 Problem Formulation

2.1 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*. Each of the users arriving into the system belongs to one of the finitely many classes. We denote the set of all classes by $\mathbb{C}$. We assume that for each state $s \in \mathbb{S}$, the number of calls of class $c \in \mathbb{C}$ arriving in any epoch is a Poisson random variable with mean $\lambda_{s,c}$. We also assume that for each call of class $c \in \mathbb{C}$ arriving in state $s \in \mathbb{S}$, the call-holding time is a geometric random variable with mean $\alpha_{s,c}$. 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. Also, our assumption that the call-holding times are declared on arrival is well-justified in the bandwidth-trading setting, as the bandwidth is typically sold for a pre-determined period of time.

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

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

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 belonging to class $c \in \mathbb{C}$ are assumed to purchase bandwidth according to a class-specific *demand function* $D_c(\cdot)$ describing the amount of resource purchased at each price by the calls of class c. For technical reasons, we assume that the demand functions are compactly-supported, strictly-decreasing, and twice differentiable over the interval of their support. Additionally, we assume that the set of prices for which the total bandwidth B and the values of the demand functions at that price are rationally related is of measure zero. Formally, define the set $\mathfrak{P}$ of prices as

$$\mathfrak{P} = \Bigg\{ p \in \mathbb{R}_+ : \text{if } \alpha_0, \alpha_c, c \in \mathbb{C} \text{ are rationals, then} \\ \alpha_0 B + \sum_{c: D_c(p) \neq 0} \alpha_c D_c(p) = 0 \Rightarrow \alpha_0, \alpha_c = 0 \Bigg\}.$$

Then we assume that the set $\mathbb{R}_+ \backslash \mathfrak{P}$ has measure zero. Usually this will hold true if the demand functions corresponding to different classes are not ratio-

nal multiples of each other over a set of non-zero measure and are strictly decreasing. For example, any countable set of *different* demand functions of the form $-\log(p/k)/k$ satisfies this property. Another example of demand functions satisfying this property is those of the form $k_1/x^{k_2} - k_3$, with appropriate values of k_1, k_2, and k_3. The demand function of a user 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. Let $\mathbb{I}$ denote the set of all the newly arrived users. Let p be the price at the current decision epoch. Let $\tilde{B}$ denote the available bandwidth in the system. Then the resource allocator allocates the bandwidth using the following algorithm.

(1) Let $i = \arg\min_{\mathbb{I}} D_{\boldsymbol{c}_i}(p)$. If more than one call belonging to the same class satisfies this relation, choose one of the calls arbitrarily. If calls belonging do different classes satisfy this relation, choose a class according to some pre-determined fixed order, and select an arbitrary call from that class.
(2) Let $b = \min\{D_{\boldsymbol{c}_i}(p), \tilde{B}/|\mathbb{I}|\}$. Allocate bandwidth b to call i.
(3) Set $\tilde{B} := \tilde{B} - b$, and $\mathbb{I} := \mathbb{I} \backslash i$.
(4) If $\mathbb{I}$ is not empty, go to step 1.

The network administrator charges each of the users for the bandwidth it has allocated to that particular user. 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.

To illustrate the concept of a demand function, we refer the reader to the utility function theory (see [15]). Utility functions are widely used in the networking and communications literature to denote the satisfaction that a user gets by consuming a particular amount of resource [9][12][13][22]. A utility function is a non-decreasing function of the amount of the commodity (in our problem, the bandwidth) being consumed, and it is common practice to assume the function to be concave (according to the law of diminishing returns) [13][21][22]. If, in addition, we assume the utility function to be strictly concave (i.e., "locally non-satiable") and thrice differentiable in the bandwidth being consumed, then the resulting demand function is twice-differentiable and strictly-decreasing, as shown below.

Let $\mathcal{U}(\cdot)$ be the utility function of a user; i.e., $\mathcal{U}(b)$ is a measure of the satisfaction that the user gets by consuming bandwidth b per-unit-time. The user demands the amount of resource that maximizes the difference between the cost it pays for the purchase and the utility it gains from the purchase. Thus, if the bandwidth is priced at p, then the user solves the following optimization

problem:

$$\text{maximize} \quad \mathcal{U}(b) - bp, \tag{2}$$

where $b \in \mathbb{R}_+$ is the decision variable.

Because we assume the utility function $\mathcal{U}(\cdot)$ to be strictly concave, there is a unique maximizer that solves (2). Thus, given p, the bandwidth demand of the user can be determined uniquely—this defines the user's demand function $p \mapsto D(p)$. Also, it can be verified that if the utility function $\mathcal{U}(\cdot)$ is thrice differentiable, then the corresponding demand function $D(\cdot)$ is twice differentiable.

As described above, when combined with the information about the available resource and the number of new users of each class in the system, the demand functions $D_c(\cdot), c \in \mathbb{C}$ uniquely tell us the amount of resource the user receives. We call this amount the *effective demand function* of the user and denote it by $\mathcal{D}_c(p, r, \vec{n})$, where $p \in \mathbb{R}_+$ is the price, $r \in \mathbb{R}_+$ is the available resource, and $\vec{n} \in \mathbb{Z}_+^{|\mathbb{C}|}$ is the vector of number of users of all the classes 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. To be precise, the network administrator solves the following optimization problem:

$$\max \quad \lim_{H\to\infty} E\left[\frac{1}{H}\sum_{k=0}^{H-1}\sum_{i\in\mathbb{Z}_+}(\boldsymbol{A}_i(k)b_i p(\boldsymbol{a}_i))\right], \tag{3}$$

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

To tackle such problems, a heuristic called the *rolling-horizon* approach is widely used [10]. 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

$$\max \quad E\left[\sum_{k=t}^{t+H-1}\sum_{i\in\mathbb{Z}_+}(\boldsymbol{A}_i(k)b_i p(\boldsymbol{a}_i))\right]. \tag{4}$$

The pricing schemes that we introduce in Section 3 use this framework for computing the optimal prices.

In this section, we formulate the pricing problem as a POMDP. To characterize the problem as a POMDP, we need to define the state space, the action space, the transition law, the reward structure, the observation space, and the conditional probability of the observations conditioned on the states. Even for our simple problem, the exact specification of these entities necessarily entails additional (sometimes complicated) notation.

State Space: We denote the state space by $\mathbb{X}$. A state $x \in \mathbb{X}$ is a triple $\langle s_x, n_x, o_x \rangle$, where $s_x \in \mathbb{S}$ gives the traffic-process state, and n_x and o_x are multisets of active user descriptions, as follows. An active user description is a triple $\langle r, b, c \rangle$ of a natural number $r > 0$, a real number $b \in [0, B]$, and a class identifier $c \in \mathbb{C}$. The user description $\langle r, b, c \rangle$ represents a user of class c consuming bandwidth b per time epoch for the remaining duration of activity r. The set n_x represents the users that have just arrived, and we require that each member $\langle r, b, c \rangle$ of n_x has $b = 0$ because 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. To be physically meaningful, the price has to be greater than or equal to zero. Thus, our action space $\mathbb{A}$ is just $\mathbb{R}_+$.

Transition Law: Let $x \in \mathbb{X}$ be the state of the system, and let $u \in \mathbb{A}$ be the action taken by the network administrator. To simplify the notation, define

$$\mathcal{L}(x) = B - \sum_{\langle r,b,c \rangle \in o_x} b,$$

and a vector $\vec{\mathcal{N}}(x)$, whose k^{th} component is given by

$$\mathcal{N}^k(x) = |\{\langle r, b, c \rangle \in n_x : c = k\}|.$$

Thus, $\mathcal{L}(x)$ is the available bandwidth at state x, and $\vec{\mathcal{N}}(x)$ is the vector of number of arrivals of each class at state x. As our state space $\mathbb{X}$ is uncountable, the transition law needs to be specified in the form $q_{\mathbb{X}|\mathbb{X}\mathbb{A}}(\mathcal{F}|(x, u))$, for all elements $\mathcal{F}$ of an appropriate σ-algebra of $\mathbb{X}$. We will specify $q_{\mathbb{X}|\mathbb{X}\mathbb{A}}(\cdot|(\cdot, \cdot))$ over the power set of $\mathbb{X}$ (the largest possible σ-algebra of $\mathbb{X}$); we can do this because if we fix (x, u), then the transition distribution $q_{\mathbb{X}|\mathbb{X}\mathbb{A}}(\cdot|(x, u))$, is characterized by a countable number of atoms.

We say that a state $y = \langle s_y, n_y, o_y \rangle$ is *reachable* from state $x = \langle s_x, n_x, o_x \rangle$ under action u if

$$o_y = \{\langle r, b, c\rangle : \langle r+1, b, c\rangle \in o_x, r > 0\} \\ \cup \left\{\langle r, \mathcal{D}_c(u, \mathcal{L}(x), \vec{\mathcal{N}}(x), c\rangle : \langle r+1, 0, c\rangle \in n_x, r > 0\right\}.$$

Let $\mathbb{Y}(x,u)$ be the set of all states reachable from state x under action u. It can easily be seen that $\mathbb{Y}(x,u)$ is at most countable. For all $s \in \mathbb{S}$ and $c \in \mathbb{C}$, define $\beta_{s,c} = 1/(1+\alpha_{s,c})$, where $\alpha_{s,c}$ is the mean duration of the calls of class c arriving in traffic state s as described in Section 2.1. Thus, $(1-\beta_{s,c})\beta_{s,c}^r$ is the probability that a call of class c arriving in traffic state s will have a call duration of r. Using this definition, we define a probability mass function $P_{\mathbb{Y}(x,u)} : \mathbb{Y}(x,u) \to [0,1]$ as follows.

$$P_{\mathbb{Y}(x,u)}(y) = P(s_x, s_y) \times \prod_{c\in\mathbb{C}} \frac{e^{-\lambda_{s_y,c}}\mathcal{N}^c(y)}{\mathcal{N}^c(y)!} \times \prod_{\{\langle r,b,c\rangle \in n_y\}} (1-\beta_{s_y,c}) \cdot \beta_{s_y,c}^r, \qquad (5)$$

where $P(\cdot,\cdot)$ is the probability transition matrix as defined in Section 2.1. It can immediately be verified that $P_{\mathbb{Y}(x,u)}(y)$ is the probability of transition to state y from state x under action u. Alternatively, one can write

$$q_{\mathbb{X}|\mathbb{X}\mathbb{A}}(\{y\}|(x,u)) = P_{\mathbb{Y}(x,u)}(y) \qquad \forall y \in \mathbb{Y}(x,u). \qquad (6)$$

Thus, the transition law of the system is given by

$$q_{\mathbb{X}|\mathbb{X}\mathbb{A}}(\mathcal{F}|(x,u)) = \sum_{y\in\mathbb{Y}(x,u)\cap\mathcal{F}} P_{\mathbb{Y}(x,u)}(y) \qquad \forall \mathcal{F} \in 2^{\mathbb{X}}. \qquad (7)$$

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. Then, the one-step reward $g : \mathbb{X} \times \mathbb{A} \to \mathbb{R}$ is given by

$$g(x,u) = \sum_{\langle r,b,c\rangle \in n_x} \mathcal{D}_c(u, \mathcal{L}(x), \vec{\mathcal{N}}(x))ur. \qquad (8)$$

Using this definition of one-step reward, our objective is to find the policy that maximizes the expected average reward, as defined in (3).

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 $o = \langle n_x, o_x\rangle$ 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 $\langle n_x, o_x\rangle$. Thus,

$$q_{\mathbb{O}|\mathbb{X}\mathbb{A}}(\mathcal{G}|x,u) = \begin{cases} 1 \text{ if } o = \langle n_x, o_x\rangle \in \mathcal{G} \subset \mathbb{O}, \\ 0 \text{ otherwise.} \end{cases} \qquad (9)$$

The above specifications of $\langle \mathbb{X}, \mathbb{A}, q_{\mathbb{X}|\mathbb{X}\mathbb{A}}, g, \mathbb{O}, q_{\mathbb{O}|\mathbb{X}\mathbb{A}} \rangle$ is the POMDP representation of the bandwidth market. In the next section we will convert this formulation to an equivalent, completely-observable MDP.

2.3 Completely-observable MDP (COMDP) Formulation

A POMDP can be converted to an equivalent COMDP whose state space consists of the 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*. As in the previous section, we also extend the functions $\mathcal{L}(\tilde{x})$ and $\vec{\mathcal{N}}(\tilde{x})$ to represent the leftover bandwidth and number of arrivals of each class at belief state $\tilde{x}$. These are not random because the relevant state components are fully observed. The action space for the MDP is the same as that for the POMDP. Thus, to complete the definition of the MDP, all we have to do is to specify the transition law and the reward structure.

Now, assume that the belief-state MDP is in state $\tilde{x}$ at a certain epoch. Consider a transition to the state $\tilde{y}$. Then the distribution of the components (random variables) $n_{\tilde{y}}$ and $o_{\tilde{y}}$ can easily be determined from $\tilde{x}$ using the traffic model (along the lines of (5)). The component (random variable) $\vec{I}_{\tilde{y}}$ can be determined from $\tilde{x}$ (and random variables $\vec{h}_{\tilde{y}}$ and $\vec{r}_{\tilde{y}}$) using Bayes' rule as follows:

$$I^{s'}_{\tilde{y}} = \frac{\sum_{s \in \mathbb{S}} q_{\mathbb{X}|\mathbb{X}\mathbb{A}}(\{\langle s', n_{\tilde{y}}, o_{\tilde{y}} \rangle\} | \langle s, n_{\tilde{x}}, o_{\tilde{x}} \rangle, a) \times I^s_{\tilde{x}}}{\sum_{s \in \mathbb{S}} \sum_{s'' \in \mathbb{S}} q_{\mathbb{X}|\mathbb{X}\mathbb{A}}(\{\langle s'', n_{\tilde{y}}, o_{\tilde{y}} \rangle\} | \langle s, n_{\tilde{x}}, o_{\tilde{x}} \rangle, a) \times I^s_{\tilde{x}}} \tag{10}$$

This, in turn, can be used to compute the transition law and the reward structure for the completely observable MDP in the belief-state space.

3 Pricing Schemes

3.1 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 [18][23], no such formulas exist for general traffic models. Here we present a stochastic gradient-ascent scheme to solve the problem in (4). Without loss of generality [3], 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,

$$\mathfrak{A}[t_1, t_2] \triangleq \{i : \boldsymbol{A}_i(k) = 1 \text{ for some } k \in [t_1, t_2]\}. \tag{11}$$

Also, associated with call $i \in \mathfrak{A}[t_1, t_2]$, we define a random variable $\tilde{\boldsymbol{d}}_{i|[t_1,t_2]}$ to be the duration of that part of call i that overlaps with the interval t_1 to t_2, given by $\min\{\boldsymbol{d}_i, \boldsymbol{d}_i + \boldsymbol{a}_i - t_1, t_2 - \boldsymbol{a}_i + 1, t_2 - t_1 + 1\}$. In addition, for notational convenience, we extend the effective demand function $\mathcal{D}_{\boldsymbol{c}_i}(p, r, \vec{n})$ to a user-specific effective demand function that is specialized to the conditions for user i, written $\mathcal{D}(p, i)$ and defined by $\mathcal{D}_{\boldsymbol{c}_i}(p, \mathcal{L}(\tilde{\boldsymbol{X}}_{a_i}), \vec{\mathcal{N}}(\tilde{\boldsymbol{X}}_{a_i}))$, where $\tilde{\boldsymbol{X}}_k$ is the system belief state at time k.

Now in this notation, setting $t = 0$ in (4), 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:

$$\boldsymbol{V}_H^{flat}(p) = \sum_{i \in \mathfrak{A}[0,H-1]} \mathcal{D}(p, i) \times p \times \tilde{\boldsymbol{d}}_{i|[0,H-1]} \ , \tag{12}$$

where H is the finite horizon. As our following result indicates, there exists a value of p that maximizes $E\boldsymbol{V}_H^{flat}(p)$.

Theorem 1 *The function $E\boldsymbol{V}_H^{flat}(\cdot)$ is continuous over $\mathbb{R}_+$, and there exists $p_0 \in \mathbb{R}_+$ that maximizes $E\boldsymbol{V}_H^{flat}(\cdot)$.*

Proof: See Appendix A. □

Our aim is to find the price $p_0 \in \mathbb{R}_+$ that maximizes $E\boldsymbol{V}_H^{flat}(p)$. The following theorem is useful in estimating the derivative of $E\boldsymbol{V}_H^{flat}(p)$ with respect to p.

Theorem 2 *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 $\partial E[\boldsymbol{V}_H^{flat}(p)]/\partial p$.*

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

Proof: See Appendix B. $\square$

Theorem 2 implies that we 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, with a step size of η_k in iteration k (using a standard stochastic approximation algorithm [7][8]). 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 [11][7][8]. The unbiasedness of the derivative estimate guarantees convergence of the algorithm to a local minimizer under general conditions (see [7][8]).

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 $0 \leq L < H$, recursively as

$$\begin{aligned}
\boldsymbol{v}_0(p) &= \sum_{\{i \in \mathfrak{A}[0,0]\}} \left(\mathcal{D}(p,i) \times p \times \tilde{\boldsymbol{d}}_{i|[0,H-1]} \right), \\
\boldsymbol{v}_L(p) &= \sum_{\{i:\boldsymbol{a}_i = L\}} \left(\mathcal{D}(p,i) \times p \times \tilde{\boldsymbol{d}}_{i|[0,H-1]} \right) + v_{L-1}(p).
\end{aligned} \tag{13}$$

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 (13), we get

$$\frac{\partial \boldsymbol{v}_L(p)}{\partial p} = \frac{\partial \boldsymbol{v}_{L-1}(p)}{\partial p} + \sum_{\{i:\boldsymbol{a}_i = L\}} \left(\mathcal{D}(p,i) + p \times \frac{\partial \mathcal{D}(p,i)}{\partial p} \right) \times \tilde{\boldsymbol{d}}_{i|[0,H-1]}. \tag{14}$$

To facilitate the development, we introduce some new terminology. Given the resource-allocation mechanism in Section 2.1, we say call i is *congested* if it receives bandwidth less than $D_{\boldsymbol{c}_i}(p)$ (the amount it would ideally purchase). Otherwise we say that the call is not congested. As all calls belonging to the same class have the same demand function, a call belonging to class c is congested if and only if all the other calls belonging to class c arriving in the same epoch are also congested (the algorithm described in Section 2.1 ensures that calls belonging to the same class receive equal amount of bandwidth). Now, at any state $\tilde{\boldsymbol{X}}$, we divide the newly arrived calls in to two sets, $\mathfrak{C}(\tilde{\boldsymbol{X}})$ and $\mathfrak{NC}(\tilde{\boldsymbol{X}})$, containing the congested and non-congested calls in state $\tilde{\boldsymbol{X}}$ respectively. Using this notation, we have

$$\frac{\partial \mathcal{D}(p,i)}{\partial p} = \begin{cases} \frac{\partial D_{\boldsymbol{c}_i}(p)}{\partial p} & \text{if not congested,} \\ \frac{1}{|\mathfrak{C}(\tilde{\boldsymbol{X}})|} \left[\frac{\partial \mathcal{L}(\tilde{\boldsymbol{X}}_{\boldsymbol{a}_i})}{\partial p} - \sum_{j \in \mathfrak{NC}(\tilde{\boldsymbol{X}})} \frac{\partial \mathcal{D}(p,j)}{\partial p} \right] & \text{otherwise.} \end{cases} \tag{15}$$

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

$$\mathcal{L}(\tilde{\boldsymbol{X}}_k) = \mathcal{L}(\tilde{\boldsymbol{X}}_{k-1}) - \sum_{\{i:\boldsymbol{a}_i=k-1\}} \mathcal{D}(p,i) + \sum_{\{i:\boldsymbol{a}_i+\boldsymbol{d}_i=k\}} \mathcal{D}(p,i), \tag{16}$$

which on differentiation gives,

$$\frac{\partial \mathcal{L}(\tilde{\boldsymbol{X}}_k)}{\partial p} = \frac{\partial \mathcal{L}(\tilde{\boldsymbol{X}}_{k-1})}{\partial p} - \sum_{\{i:\boldsymbol{a}_i=k-1\}} \frac{\partial \mathcal{D}(p,i)}{\partial p} + \sum_{\{i:\boldsymbol{a}_i+\boldsymbol{d}_i=k\}} \frac{\partial \mathcal{D}(p,i)}{\partial p}. \tag{17}$$

Equations (15) and (17) are discrete-time causal equations that can be implemented in an online fashion to compute the respective derivatives, which when combined with (14) can be used to compute $\partial \boldsymbol{V}_H^{flat}(p)/\partial p$ online. This algorithm, because of its incremental nature, computes the derivative efficiently. We tune the flat-pricing algorithm using this algorithm so that it would have a "fair chance" when we compare its performance with the two pricing mechanisms we are about to present.

3.2 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. At each epoch, it chooses a component of this vector, and charges accordingly. Because the traffic process is not fully observable, the choice of this component is based on the probability-distribution vector $\vec{I}_{\tilde{x}}$. Specifically, we can use one of the following two methods.

(1) Let $\tilde{x}$ be the state of the belief-state MDP at any decision epoch. Let $s = \arg\max_{s' \in \mathbb{S}} I_{\tilde{x}}^{s'}$. Then the network administrator charges price θ^s in that decision epoch.
(2) Let $\tilde{x}$ be the state of the belief-state MDP at any decision epoch. The network administrator generates a state s randomly according to the distribution described by $\vec{I}_{\tilde{x}}$, and charges price θ^s for that decision epoch.

In either case, we can view the vector $\vec{\theta}$ as a design parameter. A price to be charged at any decision epoch is a component of this vector. The index of the component depends on the actual realization of the random experiment. Note that once one of the above two methods has been chosen, the index selected to charge a price at any decision epoch can be determined by observing the

sample path (we extend the definition of our random experiment to encompass the random samples being drawn by the network administrator). Thus, we have a stochastic process that takes values from the set of traffic states $\mathbb{S}$. We denote this stochastic process by $\tilde{\boldsymbol{S}}(\cdot)$. Define

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

Then $E[\boldsymbol{V}_H^{reactive}(\cdot)]$ is our objective function. As the following result indicates, there exists a $\vec{\theta}_0 \in \mathbb{R}_+^{|\mathbb{S}|}$ that maximizes this objective function.

Theorem 3 *The function $E\boldsymbol{V}_H^{reactive}(\cdot)$ is continuous over $\mathbb{R}_+^{|\mathbb{S}|}$, and there exists $\vec{\theta}_0 \in \mathbb{R}_+^{|\mathbb{S}|}$ that maximizes $E\boldsymbol{V}_H^{reactive}(\cdot)$.*

Proof: The proof follows exactly same argument as that of Theorem 1. □

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

Theorem 4 *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. Moreover, $\partial \boldsymbol{V}_H^{reactive}(\vec{\theta})/\partial\theta^s$ is an unbiased estimate of $\partial E[\boldsymbol{V}_H^{reactive}(\vec{\theta})]/\partial\theta^s$ for all $s \in \mathbb{S}$.*

Proof: The proof follows exactly same argument as that of Theorem 2. □

Theorem 4 is an analog of Theorem 2 for the reactive-pricing scheme. Using this theorem, an on-line, incremental gradient-based algorithm for estimating the gradient of $E[\boldsymbol{V}_H^{reactive}(\cdot)]$ can be developed. Such an algorithm works 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 (13) to (17).

$$\begin{aligned}
\boldsymbol{v}_0(\vec{\theta}) &= \sum_{\{i \in \mathfrak{A}[0,0]\}} \left(\mathcal{D}(\theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)}, i) \times \theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)} \times \tilde{\boldsymbol{d}}_{i|[0,H-1]} \right), \\
\boldsymbol{v}_L(\vec{\theta}) &= \sum_{\{i : \boldsymbol{a}_i = L\}} \left(\mathcal{D}(\theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)}, i) \times \theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)} \times \tilde{\boldsymbol{d}}_{i|[0,H-1]} \right) + v_{L-1}(\vec{\theta}).
\end{aligned} \tag{18}$$

$$\begin{aligned}
\frac{\partial \boldsymbol{v}_L(\vec{\theta})}{\partial\theta^s} = &\frac{\partial \boldsymbol{v}_{L-1}(\vec{\theta})}{\partial\theta^s} \\
&+ \sum_{\{i : \boldsymbol{a}_i = L\}} \left(\mathcal{D}(\theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)}, i) \frac{\partial \theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)}}{\partial\theta^s} + \theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)} \frac{\partial \mathcal{D}(\theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)}, i)}{\partial\theta^s} \right) \tilde{\boldsymbol{d}}_{i|[0,H-1]}.
\end{aligned} \tag{19}$$

$$\frac{\partial \mathcal{D}(\theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)}, i)}{\partial \theta^s} = \begin{cases} 0 & \text{if not congested} \\ & \text{and } \tilde{\boldsymbol{S}}(\boldsymbol{a}_i) \neq s, \\ \frac{\partial D(\theta^s)}{\partial \theta^s} & \text{if not congested} \\ & \text{and } \tilde{\boldsymbol{S}}(\boldsymbol{a}_i) = s, \\ \frac{\left(\frac{\partial \mathcal{L}(\tilde{\boldsymbol{X}}_{\boldsymbol{a}_i})}{\partial p} - \sum_{j \in \mathfrak{N}\mathfrak{C}(\tilde{\boldsymbol{X}})} \frac{\partial \mathcal{D}(\theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_j)}, j)}{\partial \theta^s} \right)}{|\mathfrak{C}(\tilde{\boldsymbol{X}})|} & \text{otherwise.} \end{cases} \tag{20}$$

$$\mathcal{L}(\tilde{\boldsymbol{X}}_k) = \mathcal{L}(\tilde{\boldsymbol{X}}_{k-1}) - \sum_{\{i:\boldsymbol{a}_i = k-1\}} \mathcal{D}(\theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)}, i) + \sum_{\{i:\boldsymbol{a}_i + \boldsymbol{d}_i = k\}} \mathcal{D}(\theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)}, i), \tag{21}$$

$$\frac{\partial \mathcal{L}(\tilde{\boldsymbol{X}}_k)}{\partial \theta^s} = \frac{\partial \mathcal{L}(\tilde{\boldsymbol{X}}_{k-1})}{\partial \theta^s} - \sum_{\{i:\boldsymbol{a}_i = k-1\}} \frac{\partial \mathcal{D}(\theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)}, i)}{\partial \theta^s} + \sum_{\{i:\boldsymbol{a}_i + \boldsymbol{d}_i = k\}} \frac{\partial \mathcal{D}(\theta^{\tilde{\boldsymbol{S}}(\boldsymbol{a}_i)}, i)}{\partial \theta^s}. \tag{22}$$

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.

If we constrain all the components of $\vec{\theta}$ to be equal, then reactive pricing reduces to flat pricing. Thus, the set of all the flat-pricing policies is a subset of the set of all the reactive-pricing policies. Hence, it is obvious that an optimal reactive-pricing scheme would perform no worse than any flat-pricing scheme.

3.3 Spot Pricing

In a spot-pricing scheme, as in the reactive pricing scheme, the network administrator may change the price at every decision epoch. But here, the administrator is not bound to select the price as a deterministic function of the state estimate $\tilde{s}$. Solving the pricing problem as a POMDP as described in Section 2.2 results in an optimal spot-pricing policy. In Section 2.3, 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 [20]) cannot be applied in this case.

There are a number of heuristic techniques for solving such MDPs. Bertsekas and Tsitsiklis [3] have developed a framework of Neuro-Dynamic Programming (NDP) for solving problems with large state spaces. Marbach and Tsitsiklis

[17] have extended this framework to problems with large action spaces. But these techniques require identifying "important features" of the problem and casting the optimal solution as a continuously differentiable function of the features, which may not always be possible. Furthermore, there is no systematic way of identifying the features.

A number of sampling techniques have also been proposed and used to solve large MDPs. Policy Rollout [2], Hindsight Optimization [6], Parallel Rollout [5], etc., are examples of such techniques. Chang et al. [5] have compared such techniques for various problem settings and have empirically established that the performance of a particular technique depends on the specifics of the problem.

We will focus on the policy-rollout technique because of the simplicity of implementation and availability of an obvious "base policy" for rollout. In what follows, we will describe the general policy-rollout technique, and describe our approach to solving the pricing problem.

Consider the belief-state MDP as defined in Section 2.3. 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 the action $u = \mu_k^\pi(\tilde{x})$ at time k. Let Π be the set of all policies. For state $\tilde{x} \in \tilde{\mathbb{X}}$, define

$$V_L^\pi(\tilde{x}) = E\left[\left(\sum_{k=0}^{L-1} g(\tilde{\boldsymbol{X}}_k, \mu_k^\pi(\tilde{\boldsymbol{X}}_k))\right) \middle| \tilde{\boldsymbol{X}}_0 = \tilde{x}\right]. \tag{23}$$

Define

$$V_L^*(\tilde{x}) = \max_{\pi \in \Pi} V_L^\pi(\tilde{x}). \tag{24}$$

Then the horizon-k *Q-function* for state $\tilde{x}$ and action u is defined as

$$Q_k(\tilde{x}, u) = \left\{g(\tilde{x}, u) + E\left[V_{k-1}^*(\tilde{\boldsymbol{X}}_1)\middle|\tilde{\boldsymbol{X}}_0 = \tilde{x}, u\right]\right\}. \tag{25}$$

A well-known result from MDP theory [1] states that

$$V_H^*(\tilde{x}) = \max_{u \in \mathbb{A}} Q_H(\tilde{x}, u), \tag{26}$$

and the optimal policy (if it exists) is given by

$$\mu_k^*(\tilde{x}) = \arg\max_{u \in \mathbb{A}} Q_{H-k}(\tilde{x}, u). \tag{27}$$

In particular, for a fixed horizon H, the action

$$u^* = \mu_0^* = \arg\max_{u \in \mathbb{A}} Q_H(\tilde{x}, u) \tag{28}$$

is the optimal "current" decision in state $\tilde{x}$.

In practice, the Q-function is generally not available, and thus (28) is not directly useful. All of the sampling methods mentioned earlier rely on drawing random samples for estimating the Q-function, and then use the estimated Q-function, rather than the actual Q-function, to evaluate the current action according to (28). Thus, the sampling methods described above differ from each other in the way they estimate the Q-function.

To facilitate the discussion of the policy-rollout technique, we use the following definitions. 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. In case the base policy is optimal, the rollout policy is also optimal. For our pricing problem, we use an appropriately tuned reactive-pricing policy as the base policy. Here onwards, π_0 refers to such a base policy. The following theorem tells us that the function $Q_H^{\pi_0}(\tilde{x}, u)$ has a maximizer in its domain.

Theorem 5 *The function $Q_H^{\pi_0}(\tilde{x}, \cdot)$ is a continuous function over $\mathbb{R}_+$ for all belief states $\tilde{x}$, and there exists $u_0 \in \mathbb{R}_+$ that maximizes $Q_H^{\pi_0}(\tilde{x}, \cdot)$.*

Proof: The proof for this theorem follows argument similar to that of Theorem 1. □

The following result helps us estimate the derivative of $Q_H^{\pi_0}(\cdot, \cdot)$ with respect to u.

Theorem 6 *The stochastic function $\boldsymbol{q}_H^{\pi_0}(\tilde{x}, \cdot)$ is differentiable a.e. on $\mathbb{R}_+$ with probability 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 $\tilde{\boldsymbol{X}}_0 = \tilde{x}$.*

Proof: The proof follows exactly same argument as that of Theorem 2. □

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 described in Section 3.2. 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 in 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.

As mentioned earlier, under some general conditions, it can be proven that the policy obtained by rolling out a base policy outperforms the base policy. Even though it is not possible to verify these conditions for our pricing problem, we expect the spot-pricing scheme to outperform even the best reactive-pricing scheme, as the former is obtained by rolling out the latter.

4 Empirical Results

4.1 Evaluation Setup

In this section, we summarize the empirical results. We present the simulation results for a single-class case only, i.e., we assume that all the users arriving into the system have the same demand function. Our goal is 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 broker-mediated market in which only one vendor is present. The vendor itself acts as the broker. The only resource the vendor has for sale is the bandwidth on a single link. Without loss of generality, we assume that the maximum bandwidth available on this link is 1.

The users arrive according to a discretized Markov modulated Poison process (MMPP). Fig. 1 shows the state transition diagram for the underlying traffic-state process. Table 1 lists the base state-transition parameters (the entry in the i^{th} row, j^{th} column indicates the probability of transition from state i to state j), and Table 2 lists the base traffic parameters describing the traffic statistics. Unless mentioned otherwise, we will use this model with these parameters for the traffic-state process. We will modify some of these parameters for the purpose of evaluation.

As can be seen from the transition diagram and the probability transition matrix, States 1 and 2 have long holding times, whereas States 3 and 4 commute with each other very frequently. The rate of transition from States 1 and 2 to States 3 and 4 is very low. This is also true about the rate of transition from States 3 and 4 to States 1 and 2.

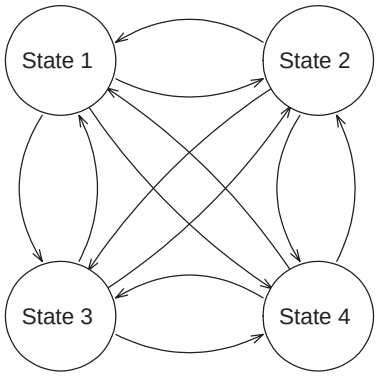

Fig. 1. State diagram for the traffic-state process

Table 1
State-transition probabilities for the traffic-state process

	State 1	State 2	state 3	State 4
State 1	0.95	0.025	0.0125	0.0125
State 2	0.00625	0.9875	0.003125	0.003125
State 3	0.025	0.025	0.55	0.4
State 4	0.025	0.025	0.4	0.55

Table 2
Arrival and holding-time parameters for the traffic-state process

	State 1	State 2	State 3	State 4
λ_s	8	0.25	2	0.25
α_s	0.875	0.875	0.75	0.5

From Table 2 it can be seen that States 1 and 3 are the states with high load, and State 2 and 4 are the states with low load. Thus States 1 and 2 together represent slowly varying load conditions, while States 3 and 4 together represent traffic in which the load conditions change abruptly.

The demand functions we use in our simulations are of the form $-\log(p/k)/k$, where p is the price of the resource, and k is a parameter. Such demand functions arise from the utility functions of the form $1 - \exp(kx)$, where x is the amount of resource being consumed. Fig. 2 shows a sample utility and a sample demand function with $k = 3$.

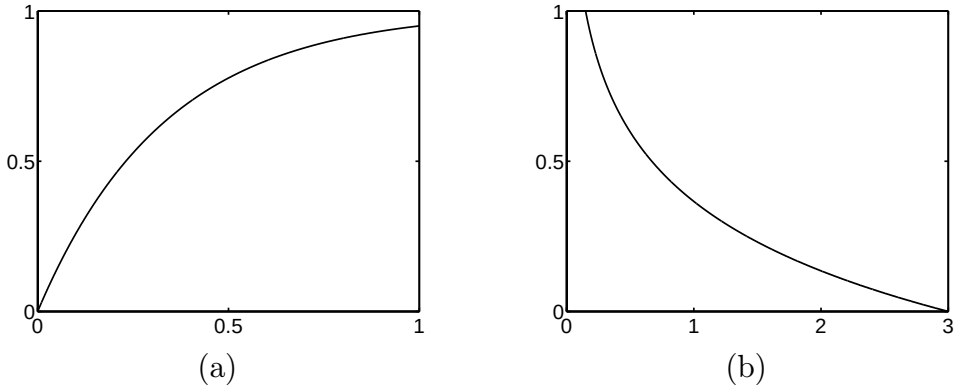

Fig. 2. (a) A sample utility function with $k = 3$, and (b) The corresponding demand function

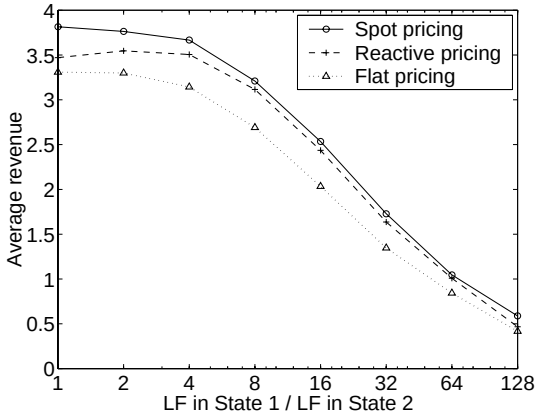

Fig. 3. Comparison of the three pricing schemes as a function of the ratio of the load factors.

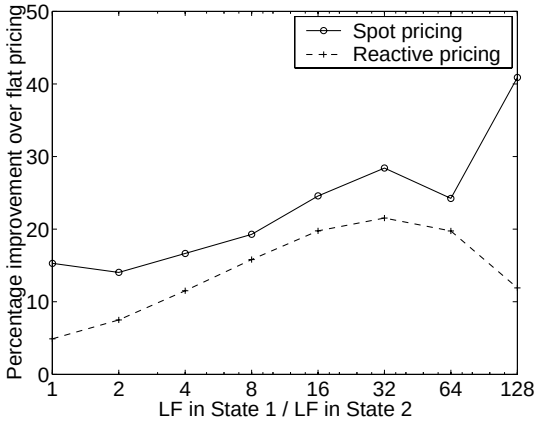

Fig. 4. Percentage improvement over the flat-pricing scheme as a function of the ratio of the load factors.

4.2 Simulation Parameters and Performance Metric

The performance of all the three schemes depends on the evaluation setup. Two of the factors affecting the evaluation setup are the traffic model and the

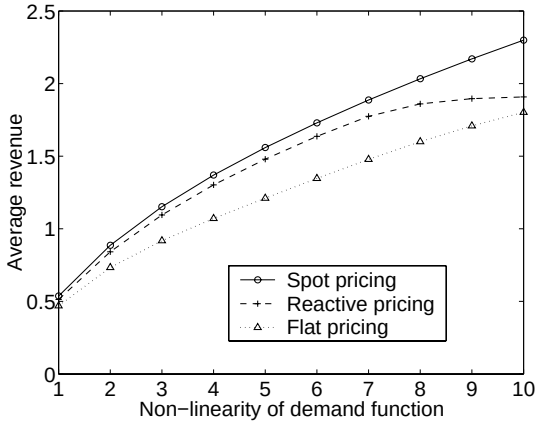

Fig. 5. Comparison of the three pricing schemes as a function of the non-linearity of the demand function.

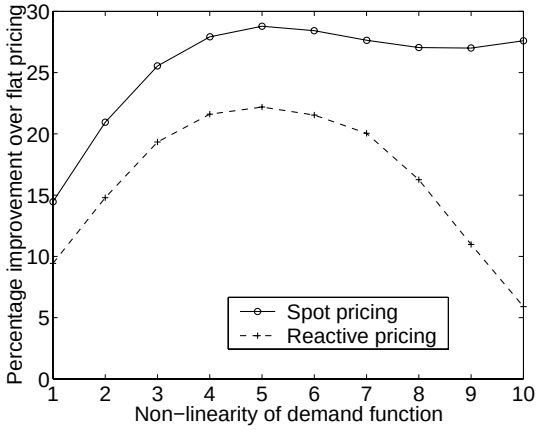

Fig. 6. Percentage improvement over the flat-pricing scheme as a function of non-linearity of the demand function.

choice of the demand functions.

As mentioned earlier, we restrict our attention to the case where all the users have the same demand function (single class) of the form $-\log(p/k)/k$. We call the parameter k the *non-linearity* of the demand function, because a function with larger value of k is more "non-linear." We will evaluate the performance of the various pricing schemes as we vary the non-linearity of the demand function.

We also wish to evaluate how the various pricing schemes cope with varying load conditions. Recall that we are using a four-state MMPP model with geometric holding times modulated by the underlying traffic state. This model naturally generates traffic with varying load conditions, if we associate a different load factor with each of the states. 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 (we drop the subscript c representing the class, as we are considering the single-class case). 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). This change in load factor has an unwanted side effect, described next.

As we change the load factor in some state, it affects the amount to which that state contributes to the overall revenue. Increasing the load factor implies that the particular state contributes more to the overall revenue and vice versa. In the extreme condition, when the load factor in a state is very high compared to that in the others, the actions taken in that state almost completely determine the net revenue. The domination of the model by a single state makes the model degenerate. But as mentioned earlier, we are interested in comparing the performance of the various pricing schemes under varying load conditions. This cannot be done if the traffic model is dominated by a single state.

To counter this effect, we alter the mean state holding time for a state in an inverse proportion to the variation in the load factor (so that the average volume of traffic served in that state remains constant). Thus, in our simulation studies, if we double the load factor in State 2 (the only state for which we modify the traffic parameters), we make the average state-holding time in that state half of what it originally was. Thus, the expected volume of calls served in State 2 is always the same as the one in the traffic model given in 1.

Our performance metric is the time average of the revenue accrued by the resource owner. In the next section, we study the performance of the three pricing schemes with respect to the above metric, and how it varies with the changes in the non-linearity of the demand function, and the changes in the load factor in State 2.

4.3 Simulation Details and Results

In this section we compare the performance of the pricing schemes based on the average revenue gathered by the resource allocator. Fig. 3 through Fig. 6 present these simulation results in a graphical form. Each of the points in this graph was generated by simulating the traffic process for 200,000 decision epochs. All the traffic parameters, except for those required for varying the load factor and state holding time in State 2, were the same as those of the base model presented in Fig. 1.

For the flat and reactive pricing methods, a gradient-based scheme was used to find the optimal prices. The gradient information was collected for 200 decision epochs, and the price vector was updated every 200 decision epochs. Recall that for defining the reactive-pricing policy, we need to specify the method of choosing a component of the parameter vector $\vec{\theta}$. We compared the performance of both the methods described in Section 3 and found that the

performance of the two schemes does not differ by a significant margin (less than 1%). For the results provided in this section, we chose the components of $\vec{\theta}$ randomly according to the belief-state distribution. For the spot pricing method, a rollout policy with a rollout horizon of 200 decision epochs was used. For estimating the gradient of the rollout reward with respect to the current action, 40 sample paths were used.

Fig. 3 and Fig. 4 compare the performance of the three traffic schemes as the ratio of the load factors 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. We can also see from Fig. 4 that at low disparity between the load factors in States 1 and 2, the percentage gain achieved by spot pricing is relatively low. This is not surprising, because as the ratio of load factors in States 1 and 2 gets close to 1, the traffic model loses its multi-state nature, and hence spot pricing loses its advantage. But it should be noted that even in the worst case, spot pricing outperforms flat pricing by about 14%. We can also see that under modest load-factor disparity between the states, reactive pricing performs very well, and its performance is comparable to that of spot pricing. But under extreme conditions (load-factor ratio very close to or very far from 1), spot pricing outperforms reactive pricing by a considerable margin.

Similar results were obtained for Fig. 5 and Fig. 6, where we compare the performance of the three pricing schemes as the non-linearity of the demand function is changed. In this case as well, it can be seen that the spot-pricing scheme always outperforms the reactive-pricing scheme, which in turn performs better than the flat-pricing scheme for all values of non-linearity. Also, from Fig. 6, it can be seen that unlike reactive pricing, spot pricing maintains its advantage even at high non-linearity values.

5 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 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] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Volumes 1 and 2*. Athena Scientific, 1995.

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

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

[4] X.-R. Cao and H.-X. Shen. Internet pricing with a game-theoretical approach: Concepts and examples. In *Proceedings of the 39th IEEE Conference on Decision and Control*, volume 3, pages 2284–2289, Sydney, 2000.

[5] H.-S. Chang. *On-line Sampling-based Control for Network Queuing*. PhD thesis, Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, July 2001.

[6] E. K. P. Chong, R. L. Givan, and H.-S. Chang. A framework for simulation-based network control via hindsight optimization. *Proceedings of the 39th IEEE Conference on Decision and Control*, 2:1433–1438, 2000.

[7] E. K. P. Chong and P. J. Ramadge. Convergence of recursive optimization algorithms using infinitesimal perturbation analysis estimates. *Discrete Event Dynamic Systems: Theory and Applications*, 1(4):339–372, June 1992.

[8] E. K. P. Chong and P. J. Ramadge. Optimization of queues using an infinitesimal perturbation analysis-based stochastic algorithm with general update times. *SIAM Journal of Control and Optimization*, 31(3):698–732, May 1993.

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

[10] O. Hernández-Lerma and J. B. Lasserre. Error bounds for rolling-horizon policies in general markov control processes. *IEEE Transactions on Automatic Control*, 35(10):1118–1124, 1990.

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

[12] F. P. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[13] S. H. Low and D. E. Lapsley. Optimization flow control, I: Basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–874, December 1999.

[14] S. H. Low and P. P. Varaiya. A new approach to service provisioning in ATM networks. *IEEE/ACM Transactions on Networking*, 1(5):547–553, October 1993.

[15] R. D. Luce and H. Raffia. *Games and Decisions*. John Wiley & Sons, Inc. New York, 1957.

[16] P. Marbach. Pricing priority classes in a differentiated services network. In *Proceedings of the 37th Annual Allerton Conference on Communication, Control, and Computing*, pages 1075–1084, University of Illinois, Urbana-Champaign, 1999.

[17] P. Marbach and J. N. Tsitsiklis. Simulation-based optimization of markov reward processes. *IEEE Transactions on Automatic Control*, 46(2):191–209, February 2001.

[18] I. C. Paschalidis and J. N. Tsitsiklis. Congestion-dependent pricing of network services. *IEEE/ACM Transactions on Networking*, 8(2):171–184, 2000.

[19] S. D. Patek and E. Campos-Náñez. Pricing of dialup services: an example of congestion-dependent pricing in the internet. In *Proceedings of the 39th IEEE Conference on Decision and Control*, volume 3, pages 2296–2301, 2000.

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

[21] H. Scarf. *The Computation of Economic Equilibria*. Yale University Press, New Haven and London, 1973.

[22] P. Thomas, D. Teneketzis, and J. K. MacKie-Mason. A market-based approach to optimal resource allocation in integrated-services connection-oriented networks. In *Proceedings of the Fifth INFORMS Telecommunications Conference*, Boca Raton, FL, 2000.

[23] Q. Wang and J. M. Peha. State-dependent pricing and its economic implications. In *Proceedings of the 7th International Conference on Telecommunication Systems Modeling and Analysis*, pages 61–71, Nashville, Tennessee, March 1999.

[24] H. Yaiche, R. R. Mazumdar, and C. Rosenberg. A game-theoretic framework for bandwidth allocation and pricing in broadband networks. *IEEE/ACM Transactions on Networking*, 8(5):667–678, October 2000.

A Proof of Theorem 1

Recall that we assume that the demand functions $D_c(\cdot), c \in \mathbb{C}$ are compactly supported. Let $p_{\max} = \sup\{p \in \mathbb{R}_+ : D_c(p) > 0 \text{ for some } c \in \mathbb{C}\}$. Now, $\boldsymbol{V}_H^{flat}(p)$ is continuous w.p. 1 on $\mathbb{R}_+$. Thus, for any $p_0 \in [0, p_{\max}]$

$$\lim_{p \to p_0} \boldsymbol{V}_H^{flat}(p) = \boldsymbol{V}_H^{flat}(p_0).$$

Also, $\boldsymbol{V}_H^{flat}(p) \leq H p_{\max} B$ w.p. 1, which in turn has finite expectation. Hence, by the Dominated Convergence Theorem,

$$\lim_{p\to p_0} E\boldsymbol{V}_H^{flat}(p) = E\left[\lim_{p\to p_0} \boldsymbol{V}_H^{flat}(p)\right]$$
$$= E\boldsymbol{V}_H^{flat}(p_0).$$

Thus, $E\boldsymbol{V}_H^{flat}(p)$ is continuous. Also, because the function is compactly supported (from 0 to $p_{\max}$), it attains its maximum for some $p_0 \in [0, p_{\max}]$. $\square$

B Proof of Theorem 2

Throughout this section, we denote the underlying probability space by $(\Omega, \mathfrak{F}, P)$. We will also assume that there exists time $-\infty < t < 0$, such that the system is empty at that time. This assumption is reasonable, because in practice the system is empty before it was started. Also, we use the notation defined in Section 3.1.

Proposition 1 *Let the price p be such that $D_c(p), c \in \mathbb{C}$ and B are not rationally related (see Section 2.1 for definition). Then $\boldsymbol{V}_H^{flat}(p)$ is differentiable w.p. 1.*

Proof: For notational convenience, we denote the portion of the bandwidth assigned to call i by $\mathcal{S}_i$. We call this portion the *slot* assigned to call i. The size of the slot (i.e., the amount of bandwidth assigned) is denoted by $|\mathcal{S}_i|$.

For the sake of contradiction, let us assume that there exists $A \subset \Omega$ such that $P(A) > 0$ and along any sample path in A, $\boldsymbol{V}_H^{flat}(p)$ is not differentiable (for notational simplicity we do not notate the dependence on the sample path). Let $\tilde{A}$ be the set of all trajectories for which $\boldsymbol{V}_H^{flat}(\cdot)$ is defined for all p and the number of users arriving in any epoch is finite. Clearly, from the assumptions about the traffic statistics, we have $P(\tilde{A}) = 1$. This in turn implies that $P(A \cap \tilde{A}) > 0$. Now, consider any trajectory in in $A \cap \tilde{A}$ for which $\boldsymbol{V}_H^{flat}(\cdot)$ is defined, and only a finitely many users arrive at any epoch. Then, for this trajectory, there exists time instances $t_j, t_j > t$ (recall that $t < 0$ is the time at which the system was empty), such that if call i arrives at time t_j, then $\mathcal{D}(p, i) = D_{\boldsymbol{c}_i}(p)$, and

$$\frac{\partial |\mathcal{S}_i|}{\partial p} \neq \frac{\partial D_{\boldsymbol{c}_i}(p)}{\partial p}.$$

Let $t_0 > t$ be first such time instant. Now, from development in Section 3.1 it follows that for this to hold we require that

$$\sum_{c\in\mathbb{C}} \mathcal{N}^c(\tilde{\boldsymbol{X}}_{t_0}) D_c(p) = \mathcal{L}(\tilde{\boldsymbol{X}}_{t_0}).$$

But because $D_c(p), c \in \mathbb{C}$ and B are not rationally related, it is evident that the calls arriving in epoch t_0 will be admitted into slots that are left vacant

by some other calls arriving between epoch t and epoch t_0. Moreover, calls of class c will be admitted only into slots left vacant by calls of the same class. Let us denote these calls by $n_i, i = 1, \cdots, \sum_{c\in\mathbb{C}} \mathcal{N}^c(\tilde{\boldsymbol{X}}_{t_0})$. But because each of the n_i arrived between epochs t and t_0, we have

$$\frac{\partial|\mathcal{S}_{n_i}|}{\partial p} = \frac{\partial D_{\boldsymbol{c}_{n_i}}(p)}{\partial p}.$$

Because each call arriving at time t_0 occupies one of the $\mathcal{S}_{n_i}$, the same holds true for such other calls as well. This is a contradiction, and hence implies that the set $A \cap \tilde{A}$ is empty, which in turn implies that $P(A) = 0$.

Thus, we have,

$$P\left(\boldsymbol{V}_H^{flat}(p) \text{ is differentiable}\right) = 1.$$

□

Proof of Theorem 2. Recall that $\boldsymbol{V}_H^{flat}(p)$ is defined as

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

By Proposition 1, $\boldsymbol{V}_H^{flat}(\cdot)$ is differentiable over $\mathbb{R}_+$, except at those p for which $D_c(p), c \in \mathbb{C}$ and B are rationally related, w.p. 1. Because we assume the set of prices for which $D_c(p), c \in \mathbb{C}$ and B are not rationally related to be of measure zero, we have that $\boldsymbol{V}_H^{flat}(p)$ is differentiable a.e. w.p. 1.

Let p be any point at which $\boldsymbol{V}_H^{flat}(p)$ is differentiable w.p. 1. Let $p_n \to p$ be any sequence in $\mathbb{R}_+$. We have,

$$\begin{aligned}
\frac{\partial E\boldsymbol{V}_H^{flat}(p)}{\partial p} &= \frac{\partial}{\partial p}\int_\Omega \boldsymbol{V}_H^{flat}(p)\, dP \\
&= \lim_{n\to\infty} \frac{\int_\Omega \boldsymbol{V}_H^{flat}(p)\, dP - \int_\Omega \boldsymbol{V}_H^{flat}(p_n)\, dP}{p - p_n} \\
&= \lim_{n\to\infty} \int_\Omega \frac{\boldsymbol{V}_H^{flat}(p) - \boldsymbol{V}_H^{flat}(p_n)}{p - p_n}\, dP
\end{aligned}$$

Now, from the definition of $\boldsymbol{V}_H^{flat}(\cdot)$, we have that $|\boldsymbol{V}_H^{flat}(p)| \le Bp$. This in turn implies

$$\left|\frac{\boldsymbol{V}_H^{flat}(p) - \boldsymbol{V}_H^{flat}(p_n)}{p - p_n}\right| \le 2B. \tag{B.1}$$

The right-hand side in (B.1) is a bounded function, which is integrable over Ω. Hence, by the Dominated Convergence Theorem,

$$\lim_{n\to\infty}\int_{\Omega}\frac{\boldsymbol{V}_H^{flat}(p)-\boldsymbol{V}_H^{flat}(p_n)}{p-p_n}\,dP=\int_{\Omega}\lim_{n\to\infty}\frac{\boldsymbol{V}_H^{flat}(p)-\boldsymbol{V}_H^{flat}(p_n)}{p-p_n}\,dP$$
$$=\int_{\Omega}\frac{\partial\boldsymbol{V}_H^{flat}(p)}{\partial p}\,dP$$

Thus the derivative and the expectation operations can be interchanged. □