# On Pricing Algorithms for Batched Content Delivery Systems

Srinivasan Jagannathan*[1] Jayanth Nayak[2] Kevin Almeroth[1] Markus Hofmann[3]

Department of Computer Science[1], Department of Electrical and Computer Engineering[2]
University of California, Santa Barbara, CA 93106-5110
{*jsrini@cs, jayanth@ece, almeroth@cs*}*.ucsb.edu*

High Speed Networks Research Department[3]
Bell Laboratories, Holmdel, NJ 07733-3030
*hofmann@bell-labs.com*

## Abstract

*Businesses offering video-on-demand (VoD) and downloadable-CD sales are growing in the Internet. Batching of requests coupled with a one-to-many delivery mechansim such as multicast can increase scalability and efficiency. There is very little insight into pricing such services in a manner that utilizes network and system resources efficiently while also maximizing the expectation of revenue. In this paper, we investigate simple, yet effective mechanisms to price content in a batching context. We observe that if customer behavior is well understood and temporally invariant, a fixed pricing scheme can maximize expectation of revenue if there are infinite resources. However, with constrained resources and potentially unknown customer behavior, only a dynamic pricing algorithm can maximize expectation of revenue. We formulate the problem of pricing as a constrained optimization problem and show that maximizing the expectation of revenue can be intractable even when the customer behavior is well known. Since customer behavior is unlikely to be well known in an Internet setting, we develop a model to understand customer behavior online and a pricing algorithm based on this model. Using simulations, we characterize the performance of this algorithm and other simple and deployable pricing schemes under different customer behavior and system load profiles. Based on our work, we propose a pricing scheme that combines the best features of the different pricing schemes and analyze its performance.*

**Keywords:** e-Content, Batching, Customer Behavior, Price, Dynamic Pricing

## 1 Introduction

The Internet is seeing an explosive growth in commercial activities. Downloadable software and multimedia are especially popular. One can think of scenarios where customers can download music, movies, and even books after online transactions. Video-on-Demand (VoD) is one such service in the Internet. However, inspite of the immense research interest in VoD over the last decade [1, 2, 3, 4, 5], commercial efforts have failed to materialize. One of the possible reasons why they have failed until now is the lack of a good business model. Given the renewed interest in such services, it is very important to develop a sound business model.

When the content is popular, and user interactivity is not required, using multicast [6, 7] or broadcast [8] mechanisms to serve many users simultaneously improves scalability of the system. This is accomplished by a technique known as *batching*. Requests for the same content are aggregated over a period of time and then served in one single transmission using a one-to-many delivery mechanism such as multicast or broadcast. This benefits the content provider greatly because fewer resources are utilized at the cost of a small waiting time for the customers. In this paper, using analysis and heuristics, we develop a business model for systems implementing batching for content distribution. In our earlier work [9, 10, 11], we have developed pricing models for systems that do not implement batching.

Pricing must take into account customer valuations as well as resource constraints. Let us consider an illustrative example. Consider a content provider selling downloadable CDs. The number of CDs that can be downloaded from the web site within a given time frame is limited by the bandwidth and server resources available. Furthermore, the resources available cannot be arbitrarily increased. This is because, the demand (or request arrival process) in an Internet setting may not be easily predictable. For instance, a very exclusive and popular music album available at the web-site may increase demand for a short period of time, say a fortnight. Once the initial popularity wanes, demand (and hence request arrival rate) will drop. Long-term investements in high capacity links and server resources to meet the demand may therefore not be a practical solution. At the same time, short-term acquisition of server resources and bandwidth may not be possible. In such a situation, two questions arise: (1) can the content provider increase revenues during the peak times by serving the same number of customers for a higher price?, and (2) can the content provider reduce the number of customers denied service[1] during peak times by charging a higher rate for the service? These are interesting questions that need to be answered for succesful deployment and acceptance

[1] We make a distinction between customers who are denied service because they do not accept the price and those who accept the price but are denied service due to resource constraints.

of content distribution networks by the commercial world.

Our objective in this paper is to answer the above questions by constructing a formal model for pricing content in a system with dynamic load patterns. To develop a thorough understanding of the fundamental problem area, we limit our considerations in this paper to a rudimentary, yet practically relevant, content delivery architecture. Our work presents essential findings, which provide the foundation for future extensions towards more complex scenarios. Even so, there are various choices for pricing the content: subscription-based pricing, quoted-price, sealed-bid auctions, etc. In this work, we restrict ourselves to a quoted-price model, wherein the content provider quotes a price to the customer. The customer may accept or reject the service based on his/her valuation of the service. We observe that if customer behavior is time invariant and well known, then a strategy of charging a fixed price can maximize the expectation of revenue if there are infinite distribution resources. When resources are constrained, fixed pricing may not maximize expectation of revenue. Nevertheless, there is usually a strong case for a fixed price because it is simple to implement. In this work, our goal is to explore the benefits of a dynamic pricing structure. We believe that subscription-based pricing coupled with a dynamic pricing scheme can address most customers' concerns. Risk-averse customers can opt for the subscription pricing while other customers can opt for the quoted-price model. Since requests are batched, subscribers' requests will not compete for resources with the quoted-price-model customers.

We formulate the problem of pricing in a batching system as a constrained optimization problem. We show that for some kinds of customer behavior, even when that behavior is well known, the problem of maximizing the expectation of revenue is intractable. In reality, customer behavior cannot be accurately known[2]. We propose a framework to understand customer behavior parameters in such a situation. Using this framework, we develop a pricing algorithm. We also study other simple, yet effective pricing schemes[3] that can be adopted in a content delivery system. Our objective in this paper is not to proclaim that one pricing mechanism is superior to another. Our aims are: (1) to understand the choices available to a content provider in a dynamic environment, and (2) to characterize these pricing options under different customer behavior and system load profiles. To this end, we perform simulations under different scenarios and evaluate the pricing schemes using two metrics: revenue and customer satisfaction (as measured in the number of requests denied due to lack of resources). Finally, we propose a hybrid scheme that combines the best features of different

[2]Market surveys usually provide some information. However, they are costly and have potentially limited accuracy in a dynamic environment like the Internet.

[3]All the pricing schemes discussed in the paper conform to the quoted-price model, where the content provider quotes a price to the customer.

pricing schemes to improve revenue and system performance.

The problem of pricing would appear to be very similar to the pricing problem at say DVD rentals or movie theaters. However, there is one significant difference that may prevent us from applying pricing strategies from such systems to a content delivery service. The crucial difference is that, all products in a content delivery system compete for *common* server and bandwidth resources. To illustrate this point, consider the case where a content provider has only enough resources to accept one request. Suppose that there are two requests–one for content $A$, where the customer is willing to pay \$5 and the other for content $B$, where the customer is willing to pay \$10. By rejecting the request for $A$ (by quoting a price greater than \$5), and accepting the request for $B$ (by quoting exactly \$10), the content provider generates more revenue. Thus, the content provider must intentionally over-price content $A$ in order to increase the returns per unit resource consumed. On the other hand, in a conventional market, it would be counter-productive to intentionally over-price any of the content. This problem is therefore specific to a system with the characteristics just described.

The rest of the paper is organized as follows. Section 2 briefly reviews related related work. Section 3 presents our system architecture and batching model. Section 4 presents our theoretical results about maximizing the expectation of revenue in a batching system. Section 5 develops a customer behavior model and a framework for estimating the parameters governing the customer behavior. Section 6 presents a class of simple and effective pricing schemes while Section 7 characterizes their behavior using simulations. In Section 8 we discuss our insights and suggest ways to extend our framework to a competitive market with multiple QoS classes. We conclude the paper in Section 9.

## 2 Related Work

Batching has primarily been studied in the context of video-on-demand (VoD). Dan et al. [2] study batching from the perspective of delay and customer loss rate. Almeroth et al. [3] propose rate-based resource allocation when using multicast for video delivery schemes. Aksoy and Franklin [12] propose a scheduling algorithm in data broadcast systems that balances content popularity and waiting time. Aggarwal et al. [13] propose a batching scheme that weights the number of requests with a factor biased against the content popularity to achieve uniformity in user loss rates. All the above do not consider the problem of pricing content. Chan and Tobagi [14] consider profit maximization in a batched VoD system with no resource constraints. They assume that the price of the content is fixed and maximize profit by changing the batching scheme. They do not focus on

how the fixed price is determined. Wolf et al. [15] also consider the problem of profit maximization for broadcasting digital goods. Their work examines scheduling the delivery of digital goods when prices, and the penalties for delayed delivery, are known. In their work too, they do not consider how the prices and the penalties are determined. Basu and Little [16] formulate the the problem of profit maximization for a VoD system in terms of price and request arrival rate. They assume that the request arrival rate is correlated with the price and that this correlation can be found using market analysis and user surveys. In a dynamic market like the Internet, where customers from geographically dispersed locations can purchase content from the same web site, market surveys may be costly and potentially inaccurate. In our earlier work [9, 10, 11], we have focussed on the problem of pricing content in a First-Come-First-Served content delivery system. Our earlier work did not consider the implications of batching. Krishnamurthy [17] investigates resource allocation for VoD based on dynamic pricing. His work also does not consider a batching system.

## 3 System Architecture and Batching Model

In a competitive market, the prices charged by competitors can affect consumer behavior and hence revenues. There has been considerable research on agents that search for the best price for a product [18, 19, 20, 21]. However, currently, not many consumers use such sophisticated mechanisms. The average Internet user is restricted to popular web sites like Yahoo! or AOL. In other words, brand loyalty is high in the Internet market[4]. Though our ultimate objective is develop pricing mechanisms for a competitive market, in this paper, we restrict ourselves to a market where there is a single content provider providing the service. Further we assume that the all customers receive the same quality-of-service (QoS). In our discussion, we explain how our framework can be extended to competitive markets with multiple QoS classes [22].

Figure 1 illustrates our content delivery architecture with an origin server and a mirror server. Customers requesting content can be redirected to the closest source. Since the available resources are different at different sources, pricing is done locally at each mirror server and at the origin server independent of the prices at other locations. Thus, in Figure 1, for receivers in sets 1 and 2, the prices are determined by the origin server, while for receivers in set 3, the prices are determined by the mirror server. Since the prices are determined locally at each source, we shall henceforth, without loss of generality, deal with a single source system. In our model, we assume that once the content provider makes the initial infrastructural investment, there are either negligible or fixed costs in

[4]This may not be true in the future where consumers can be expected to be more Internet savvy.

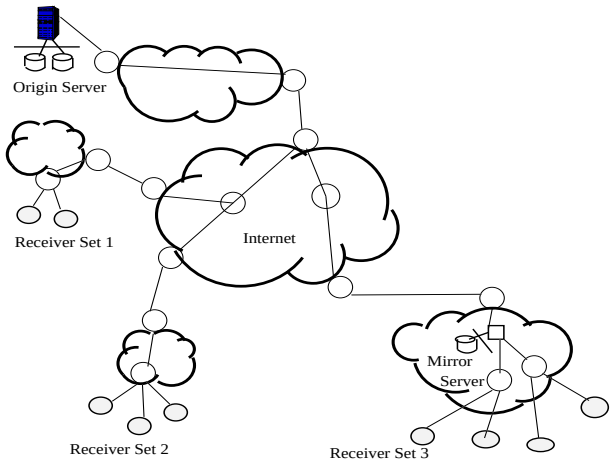


Figure 1. A Content Delivery Architecture

maintaining the resources (caches, servers, bandwidth, etc.), i.e., there are no additional costs based on the number of requests served. This is a reasonable assumption because servers incur fixed costs and bandwidth is typically bought at a flat rate. If maintenance costs are negligible or fixed, profit maximization is equivalent to revenue maximization.

The resources at each source are modelled as *logical channels*. All channels are equal and consume the same amount of resources, i.e, all users get the same quality-of-service. Suppose a new request for product $P_i$ arrives at time $t_0$. The content provider quotes a price to the customer and announces the *maximum delay* in serving that request. This delay is the batching interval, $T_i$, for that product. In our model, the maximum delay for a product, once chosen, is not varied. If the customer accepts the quoted price, and there are no previously batched requests for the same content, a new channel is reserved to serve that request. The channel is actually allocated only after the batching interval elapses at time $t_0 + T_i$. If however, there is a previously batched request for the same product, then the new request is aggregated with the older request, and the channel is allocated after the batching interval for the old request ends, at some time earlier than $t_0 + T_i$. By making the batching interval for a product invariant, we guarantee that a request is served within the maximum delay. However, when request arrival rates are exceptionally high, it is possible that there is no previously batched request and no channel available within the maximum delay. The customer is denied service and fully refunded in such cases. A critical design goal therefore is to minimize denials of service due to resource constraints.

## 4 Analytical Framework

In this section, we develop a framework for analyzing pricing in a batching system. The primary focus of this framework is maximizing the expectation of revenue. Consider an arbitrary customer who wants to purchase service $P_i$. We denote his/her decision to purchase the service by the random variable $D_i$ which can take two values, 1 for accept and 0 for reject. We begin our analysis by arguing that there exists some constant price which maximizes the expectation of revenue per customer for any given product. The proof assumes that at high prices, the probability that a customer will purchase the product is zero. This is a reasonable assumption since every customer has only a finite capacity to pay. This means that the expectation of the decision to purchase product $P_i$, given price ($p_i$), denoted by $E[D_i \mid p_i]$, is zero for large values of the price ($p_i$).

**Theorem 1** *Consider an arbitrary product $P_i$. If the expectation of the decision to buy $P_i$ given price $p_i$, ($E[D_i \mid p_i]$) is zero for all prices greater than $p_\infty$, $p_\infty > 0$, then the expectation of revenue per customer, $E[\gamma_i]$, is maximum for some constant price $p_{max}$.*

*Proof Outline:* We shall assume that $E[D_i \mid p_i]$ is continuous in the interval $p_i \in [0, \infty)$. Let the random variable $\gamma_i$ denote the revenue per customer. Suppose that different prices are charged with a probability density $f_{p_i}$. Then the expectation of revenue per customer for $P_i$ is given by:

$$\begin{aligned} E[\gamma_i] &= \int_0^\infty p_i E[D_i \mid p_i] f_{p_i}(p_i) dp_i \\ &= \int_0^{p_\infty} p_i E[D_i \mid p_i] f_{p_i}(p_i) dp_i \\ &+ \int_{p_\infty}^\infty p_i E[D_i \mid p_i] f_{p_i}(p_i) dp_i. \end{aligned} \tag{1}$$

Since $E[D_i \mid p_i]$ is zero for all $p_i$ greater than $p_\infty$, the second term in the integral defined above is zero. Since $E[D_i \mid p_i]$ is continuous in the interval $[0, p_\infty]$, there exists some $p_{max}$ in this interval at which the function $p_i E[D_i \mid p_i]$ is maximum. If there are many such points, we arbitrarily choose one of them. The expectation of the function $p_i E[D_i \mid p_i]$ is maximized if the probability density at $p_{max}$ is the highest. This will be the case when $f_{p_i}$ is the Dirac delta function $\delta(p_i - p_{max})$. In other words, the expectation of revenue is maximized when $p_i$ has a constant value $p_{max}$. □

Theorem 1 tells us that if resources are infinite, then there exists a fixed price that maximizes the revenue for that product. This price can be determined if we know how the average customer

reacts to a price, i.e., we know $E[D_i \mid p_i]$ [5] for all products $P_i$. However, when resources are limited, the same fixed price may not maximize the expectation of revenue. We illustrate this using an example. Consider a hypothetical system with 3 products, 2 channels, a batching interval of 1 minute for each product, and a request arrival rate of 2 requests per minute per product. Further, assume that at the optimal price, the acceptance rate is 0.5 for each product. In effect, if the content provider quotes the optimal price, there will be 1 customer accepting the service for each product. The content provider will need to allocate 3 channels for the three products. This clearly leads to an unstable system since only 2 channels can be allocated. On the other hand, by selectively increasing the price of some of the products, the content provider can ensure that customers for only two of the products accept, but at a higher price. The customers for the third product do not accept the price. In this manner, the content provider can earn a higher revenue by changing the prices of the products. Further, only the first customer of a product needs to pay the higher price. The second customer can be charged a discounted price because serving the second request for the same product does not consume any resources. Thus, a higher revenue is earned if the price depends on the arrival rate and on whether or not there is a previously batched request for that product.

We formalize the above ideas as follows. Let the number of products being served be $m$. For product $P_i$, if there is no existing batched request, let the price charged be $p_i$. If there is an existing batched request, let the price charged be $p_i'$. Let $T_i$ be the batching interval and $\lambda_i$ the request arrival rate for product $P_i$. Then the revenue earned per unit time is given by:

$$\begin{aligned} \mathcal{R} &= \sum_{i=1}^{m} p_i \, min\left\{\lambda_i E[D_i \mid p_i], \tfrac{1}{T_i}\right\} \\ &+ \sum_{i=1}^{m} p_i' \, max\left\{0, \lambda_i E[D_i \mid p_i'] - \tfrac{1}{T_i}\right\} \end{aligned} \tag{2}$$

Resource constraints are modeled using system utilization. System utilization is the relative fraction of time for which channels are busy servicing requests. Let $n$ be the number of channels, and the $d$ the average time to serve a request[6]. Then, the system utilization, $\rho$, is defined as the ratio of the number of requests entering the system per unit time to the maximum possible serviced requests exiting the system per unit time. The mathematical expression for system utilization, when we charge a price $p_i$ (when there are no previously batched requests) for product $P_i$, is given by:

[5] It should be a continuous and well-defined function of price.

[6] We make the assumption that all products have similar service time. For other systems, only the formulation for system utilization will change. The constraint remains the same.

| Notation | Description |
|---|---|
| $P_i$ | $i^{th}$ product |
| $D_i$ | Decision to purchase $P_i$ (0 or 1) |
| $\gamma_i$ | Revenue per customer for $P_i$ |
| $p_i$ | Price of $P_i$ when there is no batched request |
| $p_i'$ | Price of $P_i$ when there is a batched request |
| $\lambda_i$ | Request arrival rate for $P_i$ |
| $T_i$ | Batching interval for product $P_i$ |
| $\mathcal{R}$ | Total revenue |
| $n$ | Number of channels |
| $d$ | Mean service time |
| $\rho$ | System Utilization |

Table 1. Symbols Used

$$\rho = \frac{d}{n} \sum_{i=1}^{m} min \left\{ \lambda_i E[D_i \mid p_i], \frac{1}{T_i} \right\} \tag{3}$$

Note that in (3), we need not consider requests that arrive when there is a previously batched request for that product. This is because such requests do not consume extra resources. The problem of maximizing the expectation of revenue in a batching system can thus be formulated as a constrained optimization problem:

$$\text{Maximize} : \mathcal{R} \tag{4}$$

$$\frac{d}{n} \sum_{i=1}^{m} min \left\{ \lambda_i E[D_i \mid p_i], \frac{1}{T_i} \right\} \leq 1 \tag{5}$$

$$\forall i,\ 1 \leq i \leq m,\ p_i, p_i' > 0 \tag{6}$$

In our model, we assume that every customer has some valuation for each product. When the quoted price is lesser than or equal to the customer's valuation for that product, then the customer accepts the price. Different customers can have different valuations for the same product. Since humans typically deal with discrete values, it is reasonable to believe that the valuations of individual customers for a product are drawn from some arbitrary discrete probability distribution. We

now show that, if customer valuations for a product are drawn from some discrete probability distribution, then the problem of maximizing expectation of revenue in a resource constrained batching system is intractable, even when the probability distribution is known.

**Theorem 2** *Consider a batching system with $n$ channels, serving $m$ products with mean service time $d$. Let the customers' valuations for product $P_i$ be drawn from a discrete probability distribution $\Pi_i$. Let request arrival rate for product $P_i$ be $\lambda_i$. Maximizing the expectation of revenue in this system is intractable.*

*Proof Outline:* Note that a constant distribution (where there is an element whose probability is 1) is a special case of an arbitrary discrete probability distribution. Consider a constant distribution for customer valuations. Let all customers have valuation $v_i$ for product $P_i$. It can be easily verified that the optimal price of product $P_i$ is either equal to $v_i$ or is greater than it.

We show that the NP-Hard 0-1 knapsack optimization problem, can be reduced to the expected revenue maximization problem when customer valuations are drawn from a constant distribution.

We transform the knapsack problem with $m$ objects and volume constraint $V$ into revenue maximization problem as follows. Given object $o_i$ with gain $g_i$ and volume $r_i$, construct product $P_i$ with batching interval $T_i = 0$, customer valuation $g_i$ and request arrival rate $\frac{r_i}{V}$. Let $d$ = 100 and $n$ = 100 in the revenue maximization problem.

An optimal solution obtained for the above revenue maximization problem can be transformed to a solution for the knapsack problem as follows. If the optimal price for product $P_i$ is equal to $v_i$ then choose object $o_i$, else discard it. The proof that the knapsack solution is optimal is straightforward, and is omitted.

The above shows that maximizing the expectation of revenue with constant customer valuations can be as hard as solving the 0-1 knapsack problem. It follows that problem of revenue maximization when customer valuations are drawn from an arbitrary discrete probability distribution is intractable. □

## 5 Modeling Customer Behavior

In Theorem 2 we have shown the intractability of the revenue maximization problem, even when the customer valuation distribution is known. In real life, the problem is harder, because

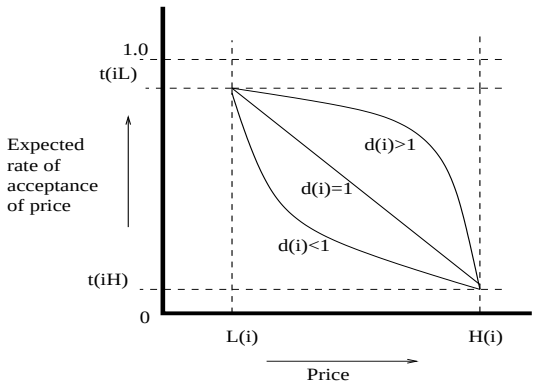


Figure 2. Non-increasing functions for expected price acceptance rate

the content-provider cannot know the customer distribution. However, one can make some general observations about customer behavior and develop a model to understand customer valuations.

We model customers as rational human beings who have a finite valuation for a product. For a rational customer population, we can infer the following about the expectation of the decision to purchase product $P_i$, given price $p_i$, $E[D_i \mid p_i]$:

- $E[D_i \mid p_i]$ is equal to 1 if price $p_i$ is 0
- $E[D_i \mid p_i]$ is a non-increasing function of price $p_i$
- There exists some price $p_{high}$ such that $E[D_i \mid p_i]$ is 0 for all prices greater than $p_{high}$
- $E[D_i \mid p_i]$ is observable, i.e., given price $p_i$, one can estimate $E[D_i \mid p_i]$ by observing how many customers accept that price out of the total number of customers who requested the product.

Based on the above observations, we propose a parametric family of non-increasing functions which can be used to approximate $E[D_i \mid p_i]$. For each product, the model has three parameters– $t_{iL}$, $t_{iH}$ and $\delta_i$. $t_{iL}$ and $t_{iH}$ are thresholds for *high* and *low* expectation of the decision to purchase. $\delta_i$ is a parameter to generate a non-increasing function whose value decreases from $t_{iL}$ to $t_{iH}$ as price of product $P_i$ increases. Prices $L_i$ and $H_i$ correspond to the prices when the expectation of the decision to purchase equals the thresholds $t_{iL}$ and $t_{iH}$ respectively. The non-increasing functions are defined as follows:

$$E[D_i \mid p_i] = \begin{cases} t_{iL} \quad , \quad 0 \leq p_i < L_i \\ (t_{iL} - t_{iH}) \left[1 - \left(\frac{p_i - L_i}{H_i - L_i}\right)^{\delta_i}\right] + t_{iH} \\ \qquad , \; L_i \leq p_i \leq H_i \\ t_{iH} \quad , \quad p_i > H_i \end{cases} \tag{7}$$

Figure 2 illustrates the function. Let us consider an example. Suppose a content provider sets a higher threshold ($t_{iL}$) of 0.95 and a lower threshold ($t_{iH}$) of 0.05. If more than 95% of the customers accept a price \$1 and fewer than 5% accept a price \$10, then $L_i$ is set to \$1 and $H_i$ is set to \$10. Now performing more observations with prices in the range $L_i$ to $H_i$, the content provider can estimate more values of the expectation of the decision to purchase. This data can be used to find the value of $\delta_i$ that minimizes the square of errors between the observed values and the approximation function. Once the approximation function is known, it can be used to solve the revenue maximization problem. Since the approximation function is continuous and differentiable[7] in the range $L_i$ to $H_i$, a numerical package can be used to find a solution[8] to the optimization problem using gradient descent methods.

## 6 Pricing Algorithms

In this section, we describe six simple algorithms. This list is by no means exhaustive. However, it covers a wide spectrum of choices in pricing on-demand services. The pricing algorithms are:

**Uniform Fixed Price (UFP):** Charge the same price for all products all the time. This algorithm can be thought of as a base case. The main problem with this algorithm is that in the absence of any information, it is difficult to choose an appropriate price.

**Product Specific Fixed Price (PSFP):** Charge a fixed price which is different for different products. This algorithm is very similar to the previous algorithm. This algorithm assumes that the content provider is able to differentiate among the products and charge a price that reflects perceived customer valuations. It is difficult to choose this price when there is lack of information.

**Time-varying Fixed Price (TFP):** Charge a fixed price for all products. Charge a different fixed

[7]For customer valuations drawn from discrete probability distributions, $E[D_i \mid p_i]$ will be a discontinuous function. But we still work with a continuous and differentiable approximation to be able to solve the revenue maximization problem.

[8]The solution however is not guaranteed to be a global optimum.

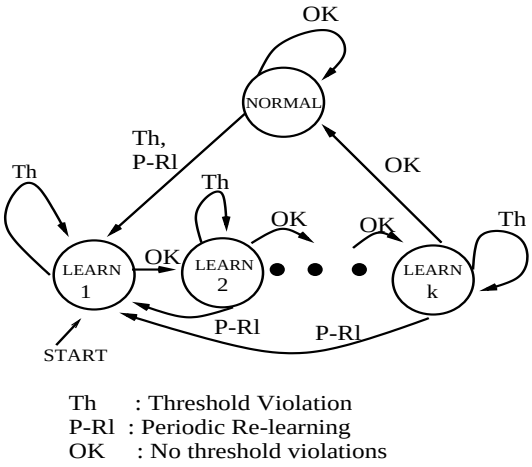


Figure 3. State Diagram for Learning Expectation of Decision to Purchase for Product $P_i$

price during "peak hours." This algorithm is similar to what is adopted at movie theatres. The price increase during peak hours can increase revenues. Again, choosing the right prices is difficult.

**System-load Price (SP):** Charge a variable price that increases exponentially with system load. This is an example of a dynamic pricing algorithm. The exponential increase in price is designed to increase revenues at times of load as well as to prevent the system from becoming unstable.

**Intelligent System-load Price (InSP):** Charge a variable price that increases exponentially with system load. If there is a previously batched request and the load is high, offer a discount. The idea behind this algorithm is that once resources have been allocated, there is no effect of an additional request on the resource consumption. Because, the exponential price will be high when the system load is high, customers may be less likely to accept the price. Therefore, a discount is offered so that more customers join and thereby increase revenue.

**Minimum Squared Error Price (MSEP):** Charge different prices for each product initially to ascertain the parameter $\delta_i$ as described in the previous section. Then solve the optimization problem to obtain "optimal" prices for each product. We now explain this algorithm in greater detail.

The revenue function depends on two quantities– request arrival rate ($\lambda_i$) and the expectation of the decision to purchase ($E[D_i \mid p_i]$). Of these, the request arrival rate can be directly observed while the expectation to purchase can be approximated using minimization of squared errors. The content provider needs to perform experiments with different prices to observe $E[D_i \mid p_i]$. Since

the number of products can be large, and request arrival rates highly skewed, we partition the set of products into classes. For instance, in a VoD system, movies can be classified as "New", "Recent", and "Old". Products are classified on the basis that customers' valuations will be similar for products in the same class. The prices for all products in a class are equal. Customer's reactions are observed for the entire class. For instance, let products $A$ and $B$ belong to the same class, say $\mathcal{S}$, and let the current price charged for both $A$ and $B$ be \$5. If during the observation period there are 10 requests for $A$ and 8 requests for $B$ and 3 customers accept the price for $A$ and 5 accept the price for $B$, then the acceptance rate for the class $\mathcal{S}$ is $\frac{3+5}{10+8} = 0.44$. Aggregating the products into classes helps in reducing the period of observation. The periods of observation are called *rounds*. When the number of requests for any product class reaches a threshold, say 100 requests, the round is terminated. The observed acceptance rates and request arrival rates are then computed for each class. The learning process for every product class follows the state diagram shown in Figure 3. There are $k$ learning states, each corresponding to one round of requests. $E[D_i \mid p_i]$ is observed for $k$ different prices. Based on these observations, the approximation function for the expectation to purchase products in that class is obtained using minimization of squared errors. The approximation function is then used in the revenue maximization problem to obtain "optimal" prices. Once these prices are obtained, there is a state transition to the "NORMAL" state. At the end of every round in the normal state, the prices $p_i$ and $p_i'$ charged in that round yield observations for $E[D_i \mid p_i]$ and $E[D_i \mid p_i']$. This data along with old data[9] is used to improve the approximation function for the expectation to purchase products in that class. Since different product classes can be in different states of the learning process at any given time, the revenue maximization is applied only on those product classes that are either exiting the "LEARN k" state or are already in the "NORMAL" state. Since customer behavior can be dynamic, it is possible that some observations violate the thresholds $t_{iL}$ or $t_{iH}$. In such cases, the threshold prices $H_i$ and $L_i$ respectively, are suitably updated and the state is returned to "LEARN 1." Since the threshold prices $L_i$ and $H_i$ can change if customer behavior changes, the states are periodically returned to the start state, "LEARN 1". Choosing the periodicity of this transition to "LEARN 1" can affect the revenue earned because all the acquired information is lost in this transition.

## 7 Simulations

We have implemented a simulator to model a content delivery system. We ran our simulations over one day of simulated time. All our simulation results are averaged over five runs with different

[9]The old data can be aged using a constant factor $\alpha$. In addition, very old observations can simply be ignored.

seed values for the random number generator. We describe the simulation scenario below.

**System Description:** We performed simulations with 100, 200, and 1000 channels. These three configurations represent highly constrained, moderately constrained, and under-constrained systems respectively. We ran simulations on these three systems to examine how the performance of the pricing algorithms varies with resource availability. We chose request service times from a uniform distribution between 90 and 110 minutes. This closely models the typical length of movies in a VoD system. We chose a fixed batching interval of 10 minutes for all products. We did not vary the batching interval because of two reasons. First, a fixed batching interval gives a service guarantee to customers. Second, the impact of the batching interval on system load and customer behavior has been examined in numerous studies [23, 3, 5]. Requests which cannot be allocated a channel within the maximum delay (batching interval) are rejected.

**Customer Choice of Products:** In all our simulations we assume that there are 100 products for the customer to choose from. Customer choice of the products was assumed to follow a Zipf-like distribution with zipf-exponent[10], $\theta = 0.73$. In a Zipf-like distribution, the $i^{th}$ popular product in a group of $m$ products is requested with probability $\frac{\frac{1}{i^\theta}}{\sum_{j=1}^{m} \frac{1}{j^\theta}}$.

**Customer Valuation Model:** We assume that the products are partitioned into classes. The valuation of a product is drawn from a probability distribution which is common for all products in a class. We further assume that the content provider knows how the products have been partitioned, but has no knowledge about the probability distribution. This is a reasonable assumption because, in real-life, the content provider can partition products into "New", "Recent", and "Old" classes. The valuations for products in one class can be expected to be significantly different from those of products in other classes. In our simulations, we chose the number of classes (say $k$) and a product was equally likely to belong to any of these $k$ classes.

Since humans typically think in terms of discrete values[11], we chose four possible discrete probability distributions for modeling customer valuations: Uniform, Bipolar, Zipf, and Constant. We briefly describe each of them below:

- **Uniform($l$, $h$, $n$):** Customer valuations are drawn from $n$ equally spaced values in the range $l$ to $h$ (both inclusive) with equal probability.

[10] Web-page accesses have been observed to obey a Zipf-like distribution with zip-exponent in the range 0.64 to 0.83 [24].

[11] In real life, customer valuations may not conform to any of these distributions. But in the absence of real life data, our objective was to test the robustness of the pricing algorithms over a range of "feasible" customer behavior patterns.

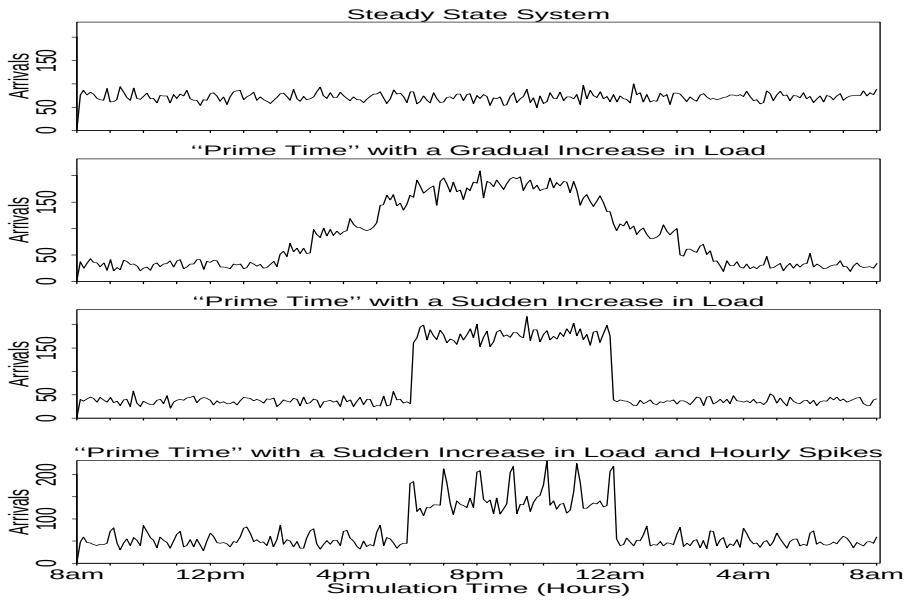


Figure 4. Workloads

- **Bipolar($l$, $h$, $r$):** Customer valuations are either $l$ with a probability $r$, or $h$ with a probability $(1 - r)$.
- **Zipf($l$, $h$, $n$, $\theta$):** Customer valuations are drawn from a set of $n$ equally spaced values in the range $l$ to $h$ (both inclusive) whose ranks follow a Zipf distribution with a zipf-exponent $\theta$. Income distributions are believed to correspond to a Zipf distribution with $\theta = 0.5$ [25].
- **Constant($c$):** All customer valuations are $c$.

In all our simulations, our unit of currency is dimes (10 dimes = \$1). We performed simulations with numerous customer valuations. In this work, we present results for customer valuations that are "realistic". We use valuations corresponding to prices charged in movies theatres. We have observed theatres charging \$2.50 for old movies, \$5.50 for matinee shows and \$8.50 for evening shows. We therefore choose three classes of products. For simplicity of labelling, we shall refer to these classes as "Old", "Recent", and "New". For the Zipf and Uniform distributions, the valuations for the classes are in the range $[20, 50]$, $[40, 70]$ and $[50, 90]$ dimes respectively. For the Bipolar distribution, the valuations for the classes are drawn from {20, 50}, {40, 70} and {50, 90} respectively. In case of the Constant distribution, the valuations for the classes are set to 30, 50 and 80 respectively.

**Pricing Policy:** We assume that the content-provider will charge at least \$1 and not more than \$10 for serving the content. We simulated all six pricing algorithms described above. In the case of static pricing policies, since we do not know what price to charge, we repeated our simulations with prices 10, 20, 30, ..., 100. We consider each of these prices equally likely and present the mean performance over this set of prices. In case of Time-varying Fixed Price (TFP) algorithm, we consider 6:00 PM (600 minutes after 8:00 AM) to 00:00 AM (960 minutes after 8:00 AM) as peak hours. For our simulations using TFP, we always assume that price during peak hours is more than the price during non-peak hours. For the Product Specific Fixed Price Algorithm (PSFP), the prices for "New" products were always more than those for "Recent" products. Similarly, the prices for "Recent" products were more than those for "Old" products.

In the case of the System-load Price (SP) algorithm, prices were computed using the function $p_i = 9 + 91^x$, where $x$ is the instantaneous system load. The instantaneous system load is computed as the ratio of the number of occupied channels to the total number of channels in the system. Using this algorithm, the customer is charged at least 10 dimes when there is no load and at most 100 dimes when the system is fully loaded. There are other functions that charge an exponentially increasing price with increasing system load. However, we consider this function as sufficiently representative of the entire class of such functions. In the case of the Intelligent System-load Price (InSP) algorithm, we do not know what discount to offer for requests when there is a previously batched request at times of high load. Whenever the instantaneous system load exceeds 0.7, we offered discounts to requests that arrived when there is a previously batched request. We repeated simulations with different discount amounts ranging from 10% to 50%. We present the mean performance assuming each of these discounts is equally likely.

**Request Arrival Process:** The arrival-rate models we used in our simulations are shown in Figure 4. These models are adapted from the work on arrival-rate based scheduling by Almeroth et al.[3]. The workloads are modeled based on a 24 hour period beginning from 8:00 AM of one day and running to 8:00 AM of the next. "Prime time" periods see a surge in demand. We have used a steady baseline workload, with no surges in demand, and three non-steady workloads. The arrival rates during prime time for the non-steady workloads were around five times greater than the normal rate. This ratio is based on statistics reported by Little and Venkatesh [26]. We simulated both gradual as well as sudden increases in arrival rate. We also used a workload with hourly spikes during prime time. This type of workload is based on the belief that the workload for some systems may be synchronized with an external event like wall-clock time. We choose the same workloads for all three systems (100, 200, and 1000 channels) so that the systems are indeed over-constrained, moderately constrained and under-constrained.

**100 Channels**

| | UFP | TFP | PSFP | SP | InSP | MSEP |
|---|---|---|---|---|---|---|
| Uniform | ⟨ 86, 0.30 ⟩ | ⟨ 89, 0.30 ⟩ | ⟨ 87, 0.28 ⟩ | ⟨ 88, 0.03 ⟩ | ⟨ 88, 0.03 ⟩ | ⟨ 119, 0.40 ⟩ |
| Bipolar | ⟨ 117, 0.31 ⟩ | ⟨ 114, 0.31 ⟩ | ⟨ 117, 0.34 ⟩ | ⟨ 113, 0.03 ⟩ | ⟨ 111, 0.03 ⟩ | ⟨ 120, 0.33 ⟩ |
| Zipf | ⟨ 47, 0.22 ⟩ | ⟨ 52, 0.22 ⟩ | ⟨ 41, 0.16 ⟩ | ⟨ 62, 0.01 ⟩ | ⟨ 61, 0.01 ⟩ | ⟨ 85, 0.27 ⟩ |
| Constant | ⟨ 110, 0.30 ⟩ | ⟨ 110, 0.31 ⟩ | ⟨ 99, 0.33 ⟩ | ⟨ 95, 0.03 ⟩ | ⟨ 91, 0.03 ⟩ | ⟨ 135, 0.29 ⟩ |

**200 Channels**

| | UFP | TFP | PSFP | SP | InSP | MSEP |
|---|---|---|---|---|---|---|
| Uniform | ⟨ 136, 0.17 ⟩ | ⟨ 144, 0.14 ⟩ | ⟨ 144, 0.13 ⟩ | ⟨ 193, 0.01 ⟩ | ⟨ 194, 0.01 ⟩ | ⟨ 220, 0.18 ⟩ |
| Bipolar | ⟨ 182, 0.16 ⟩ | ⟨ 188, 0.14 ⟩ | ⟨ 201, 0.17 ⟩ | ⟨ 218, 0.01 ⟩ | ⟨ 210, 0.01 ⟩ | ⟨ 223, 0.13 ⟩ |
| Zipf | ⟨ 78, 0.12 ⟩ | ⟨ 87, 0.09 ⟩ | ⟨ 66, 0.06 ⟩ | ⟨ 142, 0.00 ⟩ | ⟨ 137, 0.00 ⟩ | ⟨ 135, 0.14 ⟩ |
| Constant | ⟨ 171, 0.16 ⟩ | ⟨ 178, 0.14 ⟩ | ⟨ 168, 0.17 ⟩ | ⟨ 227, 0.00 ⟩ | ⟨ 202, 0.00 ⟩ | ⟨ 221, 0.14 ⟩ |

**1000 Channels**

| | UFP | TFP | PSFP | SP | InSP | MSEP |
|---|---|---|---|---|---|---|
| Uniform | ⟨ 187, 0.00 ⟩ | ⟨ 189, 0.00 ⟩ | ⟨ 198, 0.00 ⟩ | ⟨ 191, 0.00 ⟩ | ⟨ 191, 0.00 ⟩ | ⟨ 299, 0.00 ⟩ |
| Bipolar | ⟨ 234, 0.00 ⟩ | ⟨ 235, 0.00 ⟩ | ⟨ 284, 0.00 ⟩ | ⟨ 177, 0.00 ⟩ | ⟨ 177, 0.00 ⟩ | ⟨ 310, 0.00 ⟩ |
| Zipf | ⟨ 106, 0.00 ⟩ | ⟨ 109, 0.00 ⟩ | ⟨ 86, 0.00 ⟩ | ⟨ 177, 0.00 ⟩ | ⟨ 177, 0.00 ⟩ | ⟨ 194, 0.00 ⟩ |
| Constant | ⟨ 218, 0.00 ⟩ | ⟨ 220, 0.00 ⟩ | ⟨ 246, 0.00 ⟩ | ⟨ 193, 0.00 ⟩ | ⟨ 193, 0.00 ⟩ | ⟨ 313, 0.00 ⟩ |

Table 2. Simulation 1: Mean ⟨ revenue earned, denial rate ⟩ over all workloads

**Metrics:** We use two metrics in our simulations: (1) revenue earned, and (2) percentage of requests denied service because they could not be scheduled within the maximum announced delay (batching interval).

The higher the revenue earned by a pricing algorithm, the better the performance. Ideally, we would like to compare the revenues earned by each algorithm with the predicted maximum expectation of revenue, computed using complete knowledge of system and customer parameters. However, as we showed in Theorem 2, the revenue maximization problem is intractable for discrete probabilty distributions. In a system with 100 products and a variety of request arrival rates, it is difficult to compute the globally optimal revenue. We therefore only use comparison among revenues earned by the different algorithms in different scenarios to characterize them. Thus, it is quite likely that even though one algorithm performs very well, the revenue earned using that algorithm may be very far from the global optimum. Our focus therefore has been to ascertain if one of the algorithms performs *consistently* well in comparison to the other algorithms across the

different customer valuation and system load profiles.

Our other metric, the percentage of denied requests, in very important for a commercial system for two reasons. First, a high percentge of denied requests indicates that the content-provider is not living up to service guarantees. Second, it indicates that the content provider is unable to manage avaialable resources efficiently.

### 7.1 Results

We now present our simulation results. We performed two types of simulations:

1. There is no change in the customer behavior. However, the request arrivals are dynamic.
2. There is a change in the customer behavior during peak hours (6.00 PM to 9.00 PM). The request arrivals are dynamic.

To better illustrate the performance of each pricing algorithms, across different system and customer profiles, we present the results in tabular form. Each entry in the table is an ordered pair $\langle \mathcal{R}, r \rangle$. $\mathcal{R}$ is the mean revenue earned by that pricing algorithm over a number of simulations and $r$ is the mean request denial rate. By denial rate we mean the fraction of requests that could not be allocated a channel within the maximum announced delay. The revenues presented in our results have been rounded to the nearest 1000.

Note that it will not be appropriate to compare revenues within a column because, the customer valuations are different for different distributions.

**Simulation 1:** Table 2 presents the results from the first set of simulations. We observed that given a system and customer population, the performance trends did not vary significantly with workloads. We therefore present only the mean performance over all the workloads. Table 2 indicates that the MSEP algorithm gives high revenue when compared to the revenues generated by other algorithms, in all three systems, for each of the customer distributions. However, the denial rate is also among the highest for the highly constrained and moderately constrained systems. This can be attributed to the fact that the MSEP algorithm learns customer behavior by experimenting with different prices. In resource constrained systems, if a low price is charged, more customers will accept the price than can be accomodated by the system. This leads to denial of service.

SP generates revenues comparable to that of MSEP in the moderately constrained system. The reason for this is that at moderate system loads, say around 0.7 to 0.9, the price charged by

the SP algorithm is in the range 40 to 66. This price eliminates most of the requests for "Old" products while accomodating customers for the "New" and "Recent" products, most of whom have a higher valuation than the price charged. The SP algorithm therefore earns high revenue. The MSEP algorithm has a comparable performance because it learns the customer valuations. The reduction in revenue when compared to SP is probably because of the experiments performed with either too high or too low prices. The InSP algorithm does not gain much by offering discounts to requests that arrive when there is an already batched request. The reason for this is that the result presented is the mean over different discounts offered to customers. This indicates that if the "right" discount is not known, then the InSP algorithm may not be useful. In other simulations that we performed, we found that the discount that yields the highest revenue varies between 10% to 20% and depends on the workload.

In case of the fixed pricing algorithms, the result presented is the mean over several fixed prices. This is to illustrate the average case behavior when we do not know anything about the customer population. But there do exist fixed prices which yield very high revenue. For instance, in one simulation, a fixed price of 60 (using the UFP algorithm), in a system with 100 channels and a constant request arrival rate generates a revenue of 161000 when customer valuations are drawn from the uniform distribution. The MSEP algorithm was able to generate only 118800 for that simulation. However, the same fixed price, generates absolutely no revenue (revenue is zero!) when the customer valuations were drawn from the Zipf distribution. In the absence of any prior knowledge about customer valuations, choosing a fixed price can therefore be very difficult. Market surveys can possibly be used to get a rough idea what price to charge. Such surveys can also help the MSEP algorithm in limiting the range of prices over which it experiments to learn the customer behavior.

Among the fixed pricing algorithms, we noticed some interesting behavior that illustrates how different pricing in a content delivery system is as opposed to, say, pricing at a DVD rental. For the constant distribution, the customer valuations are 30, 50 and 80 respectively for each product class. For a DVD rental service with three movies with the same customer valuations, charging 30, 50 and 80 respectively (PSFP algorithm) would generate high revenue. However, in a batching system with 100 channels, and an arrival rate of 7.5 requests per minute, charging a fixed price of 80 generates around 270000 in revenue while the PSFP algorithm only generates 180000. The reason behind this is that the UFP algorithm is allocating channels only to those who value it most.

**Simulation 2:** In the second set of simulations, we varied the customer valuation during peak hours. For the results presented in this paper, all the customer valuations were increased by 30 during peak

**100 Channels**

| | UFP | TFP | PSFP | SP | InSP | MSEP |
|---|---|---|---|---|---|---|
| Uniform | ⟨ 100, 0.35 ⟩ | ⟨ 109, 0.35 ⟩ | ⟨ 103, 0.36 ⟩ | ⟨ 104, 0.04 ⟩ | ⟨ 107, 0.04 ⟩ | ⟨ 137, 0.43 ⟩ |
| Bipolar | ⟨ 130, 0.36 ⟩ | ⟨ 135, 0.37 ⟩ | ⟨ 133, 0.43 ⟩ | ⟨ 143, 0.08 ⟩ | ⟨ 136, 0.08 ⟩ | ⟨ 139, 0.37 ⟩ |
| Zipf | ⟨ 63, 0.27 ⟩ | ⟨ 70, 0.27 ⟩ | ⟨ 60, 0.24 ⟩ | ⟨ 73, 0.02 ⟩ | ⟨ 71, 0.02 ⟩ | ⟨ 106, 0.30 ⟩ |
| Constant | ⟨ 134, 0.38 ⟩ | ⟨ 152, 0.39 ⟩ | ⟨ 132, 0.46 ⟩ | ⟨ 142, 0.11 ⟩ | ⟨ 126, 0.11 ⟩ | ⟨ 169, 0.36 ⟩ |

**200 Channels**

| | UFP | TFP | PSFP | SP | InSP | MSEP |
|---|---|---|---|---|---|---|
| Uniform | ⟨ 163, 0.20 ⟩ | ⟨ 182, 0.18 ⟩ | ⟨ 176, 0.20 ⟩ | ⟨ 222, 0.01 ⟩ | ⟨ 223, 0.01 ⟩ | ⟨ 249, 0.20 ⟩ |
| Bipolar | ⟨ 207, 0.20 ⟩ | ⟨ 227, 0.18 ⟩ | ⟨ 233, 0.24 ⟩ | ⟨ 260, 0.02 ⟩ | ⟨ 247, 0.02 ⟩ | ⟨ 256, 0.16 ⟩ |
| Zipf | ⟨ 107, 0.15 ⟩ | ⟨ 120, 0.12 ⟩ | ⟨ 101, 0.12 ⟩ | ⟨ 169, 0.01 ⟩ | ⟨ 158, 0.01 ⟩ | ⟨ 171, 0.16 ⟩ |
| Constant | ⟨ 214, 0.21 ⟩ | ⟨ 251, 0.19 ⟩ | ⟨ 230, 0.28 ⟩ | ⟨ 285, 0.02 ⟩ | ⟨ 247, 0.02 ⟩ | ⟨ 283, 0.20 ⟩ |

**1000 Channels**

| | UFP | TFP | PSFP | SP | InSP | MSEP |
|---|---|---|---|---|---|---|
| Uniform | ⟨ 237, 0.00 ⟩ | ⟨ 253, 0.00 ⟩ | ⟨ 273, 0.00 ⟩ | ⟨ 196, 0.00 ⟩ | ⟨ 196, 0.00 ⟩ | ⟨ 355, 0.00 ⟩ |
| Bipolar | ⟨ 283, 0.00 ⟩ | ⟨ 303, 0.00 ⟩ | ⟨ 370, 0.00 ⟩ | ⟨ 195, 0.00 ⟩ | ⟨ 195, 0.00 ⟩ | ⟨ 354, 0.00 ⟩ |
| Zipf | ⟨ 154, 0.00 ⟩ | ⟨ 159, 0.00 ⟩ | ⟨ 151, 0.00 ⟩ | ⟨ 194, 0.00 ⟩ | ⟨ 194, 0.00 ⟩ | ⟨ 253, 0.00 ⟩ |
| Constant | ⟨ 294, 0.00 ⟩ | ⟨ 328, 0.00 ⟩ | ⟨ 393, 0.00 ⟩ | ⟨ 193, 0.00 ⟩ | ⟨ 193, 0.00 ⟩ | ⟨ 438, 0.00 ⟩ |

Table 3. Simulation 2: Mean ⟨ revenue, denial rate ⟩ over all workloads with variable customer behavior

hours. Table 3 presents the results. As in the case of the first set of simulations, we observed that the performance trends did not vary with workload. We therefore present only the mean performance over all workloads. The MSEP algorithm consistently generates high revenues in comparison to the other algorithms across all customer distributions and resource constraints, mainly because it learns the customer behavior by experimenting with different prices. And because of experimentation with different prices, it has a higher service denial rate. All the results appear consistently similar to the results in the first set of simulations. All revenues are higher than in the first set because the customer valuations are higher during the peak hours.

An interesting result in both Simulation 1 and in Simulation 2 is that the SP and InSP algorithms have minimal service denial rate. The reason for this is that when request arrivals increase, SP and InSP increase prices to such an extent that it exceeds the valuations of most customers. This reduces the number of customers accepting the service and thereby reduces the service denial rate.

Once the system returns to being lightly loaded, the price is reduced, thereby encouraging more customers to accept the price. In the case of the Constant distribution of customer valuations and system with 100 channels, we observe that the denial rate is substantially higher. This is because, at high load, the price is very high and only the customers for "New" products can accept the price. Since in a Constant distribution all customers have the same valuation, either all customers can accept the high price, or none can. This causes an oscillating behavior where many customers accept the price at the same time (and get rejected by the system) or none accept, thereby lowering the price, which again causes many to accept and so on.

We performed other simulations where we varied the frequency with which the customer behavior changed. We observed that the performance of all the algorithms decays when the customer valuations change rapidly. However, the relative performance of each of the algorithms was fairly consistent with the trends observed in Table 2 and Table 3. We therefore do not present those results.

### 7.2 A Hybrid Pricing Algorithm

We observed in the simulation results that the MSEP algorithm generates high revenue, but has an exceptionally high service denial rate. On the other hand, the SP algorithm greatly reduces the service denial rate. An interesting possibility would be to combine the best features of these two algorithms. To this end, we propose the following. When the instantaneous system load is below a threshold, use MSEP. If instantaneous system load exceeds the threshold and there is not a previously batched request, use SP. If instantaneous system load exceeds the threshold but these is an already batched request, then use MSEP. The idea behind this is that, if there is an already batched request, no extra resources are consumed, so an exponentially high price may not be suitable. The question then arises, how do we decide the treshold? We assumed different values for the threshold and ran simulations using all the scenarios. We found that though there is some variability in performance, a threshold value in the range [0.7,0.8] generates high revenues while reducing the service denial rates over all customer behavior and workload profiles. In Table 4, we present the results for Simulation 1 and Simulation 2, using the Hybrid algorithm with the system load threshold set to 0.75. Notice that the revenue of the hybrid algorithm, though slightly lower, is comparable with that of MSEP for all three systems. Also note that the service denial rate is comparable to that of SP for all three systems. Thus, for a slight loss in revenue, the service denial rate has been greatly reduced.

**Simulation 1**

| | 100 Channels | 200 Channels | 1000 Channels |
|---|---|---|---|
| Uniform | ⟨ 107, 0.03 ⟩ | ⟨ 200, 0.01 ⟩ | ⟨ 299, 0.00 ⟩ |
| Bipolar | ⟨ 140, 0.03 ⟩ | ⟨ 242, 0.01 ⟩ | ⟨ 310, 0.00 ⟩ |
| Zipf | ⟨ 72, 0.02 ⟩ | ⟨ 123, 0.00 ⟩ | ⟨ 194, 0.00 ⟩ |
| Constant | ⟨ 115, 0.03 ⟩ | ⟨ 195, 0.00 ⟩ | ⟨ 313, 0.00 ⟩ |

**Simulation 2**

| | 100 Channels | 200 Channels | 1000 Channels |
|---|---|---|---|
| Uniform | ⟨ 130, 0.04 ⟩ | ⟨ 242, 0.01 ⟩ | ⟨ 355, 0.00 ⟩ |
| Bipolar | ⟨ 163, 0.06 ⟩ | ⟨ 271, 0.01 ⟩ | ⟨ 354, 0.00 ⟩ |
| Zipf | ⟨ 85, 0.03 ⟩ | ⟨ 153, 0.01 ⟩ | ⟨ 253, 0.00 ⟩ |
| Constant | ⟨ 152, 0.10 ⟩ | ⟨ 258, 0.03 ⟩ | ⟨ 438, 0.00 ⟩ |

Table 4. Mean ⟨ revenue, denial rate ⟩ for Hybrid Algorithm

## 8 Future Work

There are a number of advanced issues that would be interesting to study in greater detail, but have not been touched upon in this work. Instead, our interest has been on a comprehensive and thorough study of the more fundamental issues. Based on these findings, we intend to explore advanved topics in the future, such as the ones outlined below.

**Quality-of-Service:** A realistic content-delivery system should have multiple classes of service. A system with multiple QoS classes can be modelled using a fluid resource model. With a single service class, one can define resource constraints by using the notion of system utilization. With multiple QoS classes however, one cannot directly apply the notion of system utilization. This is because the maximum number of requests that can be serviced by the system depends on the relative fraction of resources allocated to each QoS class. If this fraction is dynamic and depends on the rate of request arrivals for each class, system utilization cannot be defined. A heuristic that can be used is the observed relative fraction of request arrivals for each QoS class to define the system utilization. This topic is dealt with in greater detail in our subsequent work [22].

**Competitive Markets:** In this work, we have assumed that there is no competition. This could be an unrealistic assumption. However, there are many factors that suggest that the e-content market is not perfectly competitive. For instance, the content provider could be the *only* distributor of the

content. This is the case when either the content provider owns the content or has exclusive license to distribute the content. This market is clearly monopolistic. Notice however, that there may be indirect competition from "similar" content from other content providers. This will have an indirect effect on the customers' valuation of the content. Since MSEP learns these valuations, indirect competition will not significantly impact the pricing model.

The MSEP algorithm can work equally well in a competitive market if we make the following change. Suppose that the content-provider makes a policy decision to charge not more than $k\%$ of the lowest price charged for that product by any competitor. Then, one can write scripts to ascertain the current price at some of the leading competitor's websites. The lowest among the prices determined in this fashion can be used to set an additional constraint on the prices in the revenue maximization problem. Such constraints only affect the domain in which a gradient descent method will search for a solution.

**Discounts for Denied Requests:** Instead of denying requests that cannot be served within the maximum delay, the content provider can adopt an approach similar to that investigated by Wolf et al. [15]. The content provider can offer a discount for delayed delivery of the product. The problem of maximizing revenue is even harder in this case. Various heuristics need to be explored to offer the "right" discount. The customer model will also be more complex because the customer's valuation is now dependent on the delay and the discount offered.

**Impact of Dynamic Pricing on Customer Behavior:** Dynamic pricing algorithms, while generating high revenues or reducing service denial rates, may make customers uncomfortable. A subscription-pricing scheme, if coupled with a dynamic pricing scheme, can mitigate customer concerns. In a subscription based scheme, customers pay a bulk amount for making a predetermined number of requests, or alternately, pay a bulk amount for unlimited requests over a finite period of time. It would be interesting to ascertain how subscription prices are determined and if system resource constraints play a role in determining these prices.

Another approach to control the negative impact of dynamic prices on customer behavior is to offer a fixed price but a dynamic discount. For example, suppose that the content provider fixes the price at \$10. If the dynamic price suggested by MSEP is \$8, then the discount offered is \$2. Similarly, if after the next price revision, MSEP suggests a price of \$7, the discount is increased to \$3. Since the price is "fixed", the customers will still be comfortable with the dynamic prices.

**Other Pricing Models:** Sealed-bid auctions can be adapted to batched content delivery systems. The bids can be cleared once every batching interval. Strategies to choose winning bids in a manner that yields high revenue as well as increases efficiency of resource utilization is an interesting avenue

of research. Mechanisms to prevent collusion also need to be studied. Collusion prevention also needs to be studied in the context of learning algorithms like MSEP.

## 9 Summary

We examined the problem of pricing in a content delivery system impementing batching. We noted that pricing must consider customers' valuations as well as resource constraints. We observed that in the absence of any resource constraints, a fixed pricing strategy can maximize the expectation of revenue. We showed that this may not hold true when resources are limited. We formulated the problem of revenue maximization as one of constrained optimization and showed its intractability under certain conditions. Since customer behavior may not be known, we propose a class of non-increasing functions to approximate customers' expectation to purchase products and discuss how the parameters governing this function can be ascertained. We also describe a pricing algorithm (MSEP) based on this mechanism.

To better understand the dynamics of pricing, we studied the performance of six different pricing algorithms under different customer behavior and system load profiles. We discovered that the class of fixed pricing algorithms can generate high revenues, if there is some prior knowledge about customer behavior. In the absence of such knowledge, the revenue earned can be arbitrarily low depending on the customer behavior. In such situations, the MSEP algorithm yields consistently high revenues, at the expense of a high service denial rate. In contrast, a system-load based pricing algorithm (SP) has minimal service denial rate. We therefore combined the MSEP algorithm and the SP algorithm in order to generate high revenues with minimal service denial rate. We found that though there is some loss in revenue when compared with MSEP, the gains in terms of reduced service denial rate far outweigh the loss.

## References


[1] J.-P. Nussbaumer, B. V. Patel, F. Schaffa, and J. P. G. Sterbenz, "Networking requirements for interactive video on demand," *IEEE Journal of Selected Areas in Communications*, vol. 13, no. 5, pp. 779–787, 1995.

[2] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling policies for an on-demand video server with batching," in *ACM Multimedia*, (San Francisco, California, USA), October 1994.

[3] K. Almeroth, A. Dan, D. Sitaram, and W. Tetzlaff, "Long term channel allocation strategies for video applications," in *IEEE Infocom*, April 1997.

[4] X. Li, M. H. Ammar, and S. Paul, "Video multicast over the Internet," *IEEE Network Magazine*, April 1999.

[5] S. Chan and F. Tobagi, "Providing distributed on-demand video services using multicasting and local caching," in *Proceedings of IEEE Multimedia Applications, Services, and Technologies*, June 1999.

[6] K. Almeroth, "The evolution of multicast: From the MBone to inter-domain multicast to Internet2 deployment," *IEEE Network*, January/February 2000.

[7] B. Williamson, *Developing IP Multicast Networks.* Cisco Press, 2000.

[8] T. Henderson and R. Katz, "Transport protocols for Internet-compatible satellite networks," *IEEE Journal of Selected Areas in Communications*, vol. 17, pp. 345–359, February 1999.

[9] S. Jagannathan and K. C. Almeroth, "An adaptive pricing scheme for content delivery systems," in *Global Internet Symposium*, (San Antonio, Texas, USA), November 2001.

[10] S. Jagannathan and K. C. Almeroth, "The dynamics of price, revenue and system utilization," in *Management of Multimedia Networks and Services*, (Chicago, Illinois, USA), October 2001.

[11] S. Jagannathan, J. Nayak, K. Almeroth, and M. Hofmann, "A model for discovering customer value for e-content," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*, (Edmonton, Canada), July 2002.

[12] D. Aksoy and M. Franklin, "Rxw: A scheduling approach for large-scale on-demand data broadcast," in *Proceedings of IEEE Infocom*, 1998.

[13] C. C. Aggarwal, J. Wolf, and P. Yu, "On optimal batching policies for video-on-demand storage servers," in *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, 1996.

[14] S. Chan and F. Tobagi, "On achieving profit in providing near video-on-demand services," in *Proceedings of the 1999 IEEE International Conference on Communications (ICC'99)*, June 1999.

[15] J. Wolf, M. Squillante, J. Turek, and P. Yu, "Scheduling algorithms for broadcast delivery of digital products," *IEEE Transactions on Knowledge and Data Engineering*, 2000.

[16] P. Basu and T. Little, "Pricing considerations in video-on-demand systems," in *ACM Multimedia Conference*, November 2000.

[17] A. Krishnamurthy, *A Dynamic Resource Reservation and Pricing Policy for Scalable Video Delivery.* PhD thesis, Boston University, September 1995.

[18] J. Y. Bakos, "Reducing buyer search costs: Implications for electronic marketplaces," *Management Science*, vol. 43(12), December 1997.

[19] A. Chavez and P. Maes, "Kasbah: an agent marketplace for buying and selling goods," in *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, April 1996.

[20] A. Greenwald and J. Kephart, "Shopbots and pricebots," in *Sixteenth International Joint Conference on Artificial Intelligence*, August 1999.

[21] M. Tsvetovaty, M. Gini, B. Mobasher, and Z. Wieckowski, "Magma: an agent-based virtual market for electronic commerce," *Applied Artificial Intelligence*, 1997.

[22] S. Jagannathan and K. Almeroth, "Pricing and resource provisioning for delivering e-content on-demand with multiple levels-of-service," in *International Workshop on Internet Charging and QoS Technologies*, October 2002.

[23] L. Gao, J. Kurose, and D. Towsley, "Efficient schemes for broadcasting popular videos," in *Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'98)*, July 1998.

[24] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *Infocom*, pp. 126–134, 1999.

[25] G. Zipf, *Human Behavior and the Principle of Least Effort, an Introduction to Human Ecology*. Addison-Wesley, 1949.

[26] T. Little and D. Venkatesh, "Prospects for interactive video-on-demand," *IEEE Multimedia*, pp. 14–23, Fall 1994.