Pricing-Based Strategies for Autonomic Control of Web Servers for Time-Varying Request Arrivals

Yiyu Chen, Amitayu Das, Natarajan Gautam*, Qian Wang and Anand Sivasubramaniam.

The Pennsylvania State University
University Park, PA 16801

yzc107@psu.edu, adas@cse.psu.edu, ngautam@psu.edu, quw6@psu.edu, anand@cse.psu.edu

Abstract

This paper considers a web service that receives requests from customers at various rates at different times. The objective is to build an autonomic system that is tuned to different settings based on the varying conditions, both internally and externally. It has developed revenue-based pricing as well as admission control strategies, taking into account Quality of Service issues such as slow down and fairness aspects. Three heuristics are developed in this paper to address the pricing and admission control problem. The three heuristics are: 1) static pricing combined with queue-length-threshold-based admission control, 2) dynamic optimal pricing with no admission control, and 3) static pricing with nonnegative-profit-based admission control. These three strategies are benchmarked against a fourth strategy (called - do nothing) with no pricing and no admission control. The paper evaluates and compares their performance, implementability and computational complexity. The conclusion is that the web server revenue can be significantly increased by appropriately turning away customers via pricing or admission control mechanisms, and this can be done autonomically in the web server.

Keywords - Pricing, slowdown, admission control, revenue, web-server.

*Corresponding author; Address: 349 Leonhard Building, University Park, PA 16801; Phone: 814-865-1239; Fax: 814-863-4745.

1 Introduction

The motivation for this paper stems from two important emerging trends: (i) the growth of commercial services on the Internet, and (ii) the need for deploying systems that can automatically tune themselves. As business enterprises and individual users become increasingly dependent on the Internet (the world wide web in particular), there is a growing need for the deployment of commercial (web) services that can be relied upon to provide a certain Quality of Service (QoS). Even if best-effort service can be an option for certain kinds of services or at certain points in time, users may be willing to pay (even if it is only a small amount), to attain better QoS especially when there is a fear of getting badly degraded service (during periods of high load). At the same time, the service provider should be very careful in pricing the QoS so as to not only maximize the revenue earned but also to not price the service exorbitantly high that many clients get turned away [1, 2, 3]. Such pricing needs to be determined dynamically, while still making the contract between QoS and price available to the client upon request arrival, since there can be widespread fluctuations on imposed load. The need for autonomic tuning becomes particularly acute not only because of the high costs of involving humans in system operation (the Total Cost of Ownership - TCO - that several commercial organizations have recently pointed out [12, 16]) but also because such tuning should be nimble enough to adapt to workload changes. This paper explores solutions for this largely uninvestigated problem, introducing not just a pricing structure that can be dynamic and fair to both clients and servers, but also examines techniques for adaptively setting this price.

For the purposes of the evaluations in this study, this paper specifically focuses on a web service on the Internet that needs to cater to the HTTP file requests of a large number of clients on a continuous basis. It is to be noted that the main ideas presented here are applicable to many other kinds of commercial services as well. As pointed out by several other studies analyzing web server requests [4, 15], the load can be widely varying over the duration of the day. If the server charges a fixed price regardless of load, it can result in either being unfair to the clients or a loss in revenue earned. For instance, if the price is fixed (to a very high value) based on peak/high load conditions, then the clients may end up paying such a high price at even low load situations. Such a high price can also make this server less competitive, potentially making clients turn to alternate sources. At the same time, fixing the price to a low value can decrease the revenue that a server could potentially make. These arguments make it important to come up with a pricing strategy that is adaptive to varying loads, is fair to the clients, and at the same time does not significantly compromise on the server's revenue. This can be done by either adaptively varying the price or rejecting requests (and keeping price static).

The first contribution of this paper is a combination of pricing and admission control strategy for such commercial services that take the above-mentioned factors into consideration. This is essentially a 2-step process in this proposal. In the first step, the customer/client is offered a price package, which in turn has three main components: (a) a part that can be adaptively determined and is charged to each request regardless of the request's parameters, (b) a part that is a function of its service time requests (is essentially proportional to file size requested in this scenario), and (c) a part that is returned back to the consumer based on the degradation in QoS that is experienced (a penalty proportional to the slowdown that the request actually experiences compared to serving the request in isolation). In (b) and (c), this paper is trying to provide a (fixed) fair price for the desired QoS to each request. It uses (a) as a means to control the revenue earned by the server as a function of the system load. At the same time, the customer can choose to ask for service or leave based on the attractiveness of this part of the price. This part could be set to a high value during periods of high load to deter customers from joining (when the server is already earning

enough revenue), and to a lower value during low load periods (to encourage clients to join). The second step of the mechanism enforces a level of admission control imposed by the server. It can so happen that a client despite the high price, chooses to get service at a time of high load. The server can use the second step as a way of rejecting this client if it feels that the revenue it would make from this client can be offset by the lower revenue due to degradation in QoS for others. To authors' knowledge, this is the first paper to offer such a 2-level integrated pricing and admission control strategy that takes the different desirable aspects of fairness, profit and modulation to load dynamics into account.

The second set of contributions in this paper is in developing and evaluating three heuristics for adaptively modulating the above pricing/admission-control strategy to maximize revenue. Some of the factors that need to be taken into consideration when designing techniques for modulating the parameters are revenue enhancement (for the server), implementability (whether all the parameters and performance characteristics are available at the point of decision making) and computational complexity (to ensure that the decision-making/optimization is itself not very time consuming to become a deterrent to server performance). Taking these factors into account, this paper presents three heuristics for decision making at various points/granularities in time and compare them with a fourth strategy, the "do nothing" policy of admitting all customers and charging them a price that is only based on (b) and (c) mentioned above.

The rest of this paper is organized as follows. The next section gives a quick review of some related topics. Section 3 gives a more formal description of the system model. The adaptive decision-making techniques are described in section 4. Results from detailed simulations using real HTTP traces are given in section 5. Finally, section 6 summarizes the results.

2 Related Work

Usage of shared data centers are becoming very common nowadays. Web-applications are hosted on such data centers, where the application owner pays for the server resources and server provides the application guarantees on resource availability and performance. Depending on the web-application, web workload vary dynamically with time. Given this fact, online techniques are necessary to adapt to changing workloads fairly and efficiently, especially under transient overload conditions. Shenoy et al [9] proposes an online optimization-based technique for adaptive resource allocation in servers at the shared data centers. They show that their technique reacts to changing workloads by adaptively varying the resource shares of applications. They also show that such online adaptive technique is capable of reallocating resources in a better manner than static approaches. Similar problems of dynamic allocation for storage bandwidth to application classes has been addressed in [19]. It proposes a reinforcement-based learning approach to solve that. This approach is supposed to react to dynamically changing workloads and is stable under such workloads. This approach has been shown to be superior than a simple learning-based technique for storage bandwidth allocation. Likewise, on-line management of storage systems is addressed in [5].

In a constantly changing environment, a mechanism for optimizing server's operation has been proposed in [22]. There is an adaptive admission control method for offering service differentiation among various client classes. Each class is characterized by QoS performance guarantee, expressed by its service level agreement (SLA). The success of the mechanism depends on its ability to minimize the revenue loss. Resource overbooking is done after detailed application profiling for shared hosting platforms in [20]. It shows that these techniques can be combined with commonly used QoS resource allocation mechanisms to provide application isolation and performance guarantees

at run-time. It shows that the efficiency benefits from controlled overbooking of resources can be dramatic. Similar ideas for an integrated resource management framework have been proposed in [17]. It introduces the metric quality-aware service yield to combine the overall system efficiency and individual service response time in one flexible model.

Abdelzaher et al [13] presents the design and implementation of an adaptive architecture to provide relative delay guarantees for different service classes on web servers. This feedback control approach guarantees relative delays in web servers. The control-theoretic approach shows that this adaptive server achieves robust relative delay guarantees even when workload varies significantly.

Wierman and Harchol-Balter [21] define the fairness of scheduling policies in $M/G/1$ queues by the expected slow down value with respect to any job size, X, should be less than $1/(1-\rho)$, where $\rho = \lambda E(X)$. A scheduling policy is fair if it treats every job size fairly. They categorized three types of unfairness, always fair, sometimes unfair, and always unfair. All non-sized based, non-preemptive policies are always unfair, such as first-come-first-served (FCFS) policy. Based on that, in this paper, compensations for users are made by offering a price discount proportional to slowdown.

Chun and Culler [10] assume customer evaluation of system performance linearly decays with slowdown. Market-based cluster batch scheduler incorporate user evaluation as user-centric performance metrics as opposed to system-centric metrics which do not take customer utility into account and thus are not good measures of resource allocations. They demonstrated experiments in market-based FCFS and shortest job first, and concluded that market-based resource management results in significantly more value delivered to users under a variety of workloads compared to traditional approaches.

There are several studies that focus on pricing issues in networking. In particular, Paschalidis and Liu [14] consider a pricing scheme that is based on the congestion level of the network. However, they show that static pricing is (asymptotically) optimal. Further, they add that an optimal adaptive pricing policy may not be able to outperform (in terms of revenue) an appropriately chosen static price that does not depend on congestion levels. This insight is used in two of the strategies where a static price and only incorporate admission control have been considered.

3 System Model and Assumptions

Since the main objective of this paper is to illustrate the adaptive pricing and admission control schemes, a simple web service scenario with a single web server is considered in this paper. The goal of the web service is to maximize the revenue, and at the same time, to be fair to the users in terms of what they pay for their request service size and quality of service they get.

A combination of pricing and admission control strategy is proposed in this paper. The 2-level architecture is illustrated in Figure 1. The server dynamically (or statically) determines the price to charge each request. The clients will be informed the current price when they arrive. The clients can choose to accept the price and enter the system or choose to leave the system if the price is too high for them. After the clients enter the system, the server can impose a second-level admission control and reject requests to improve system performance and service profit. For example, this second-level admission control can be based on the state of the system such as the queue length (the request will be rejected if the queue length exceeds a threshold value) or based on certain profit constraints (e.g. the request will be rejected if profit becomes negative).

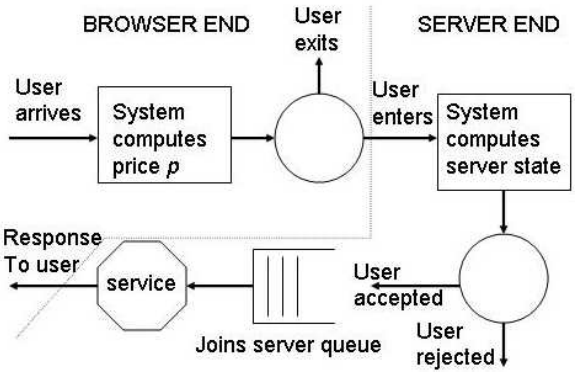

Figure 1: System Architecture for 2-level Control

The price charged on each request will affect a client to enter or leave the system and thus affect the arrival rate that really enters the system. That is, if the price is too high, there will be less clients entering the system, and if the price is too low, the server cannot make a good revenue even though they have served more clients. This pricing scheme reflects the supply/demand interaction in an economic market by incorporating the effect of price on consumers' behavior.

In this paper, a representative demand curve has been considered characterizing the relation of price (p) versus the probability (q) that an arriving customer would accept the price and enter the system. In particular, it considers a decreasing, differentiable and invertible function $f(\cdot)$ such that

$$q = f(p). \tag{1}$$

This control design and optimization schemes do not depend on the choice of $f(\cdot)$. However for numerical results, the following function is chosen $f(p) = 1 - p^2/G^2$ (where G is a constant).

Let $\overline{\lambda}$ be the customer arrival rate, then the expected number of customers that accept the price in a unit of time is $\lambda = q\overline{\lambda} = f(p)\overline{\lambda}$. Note that λ is the effective arrival rate that actually enters the system.

The web server serves the requests that enter the system in an FCFS manner. For each request, the server (i) charges a fixed price p as described as above, (ii) charges a rate a per unit time of service that the user requests (i.e. the charge of a request is proportional to the byte size of the requested file), and (iii) pays a penalty with a rate c per unit of slowdown experienced by an arrival into the system, which is defined (as in [21]) as the ratio of response time and service time. Let S be the service time requirement of a request and W be the response time experienced by the user. Then, the expected revenue gained by server from a single request is $p + aE[S] - cE[\frac{W}{S}]$. Assuming that the service time requirement is independent and identically distributed (iid), since λ is the number of customers that accept price p per unit time, the expected revenue per unit time (R) for the server is:

$$R = \lambda \left(p + aE[S] - cE\left[\frac{W}{S}\right] \right), \tag{2}$$

Notice that the arrival and service distribution are required to compute R in closed form. This paper fixes the parameters a and c as constant, and run numerical examples based on several set of values of them (see section 5). The server dynamically determines the price p to maximize the revenue and then informs clients the price p along with the parameters (a, c) on the fly.

4 Admission Control Strategies

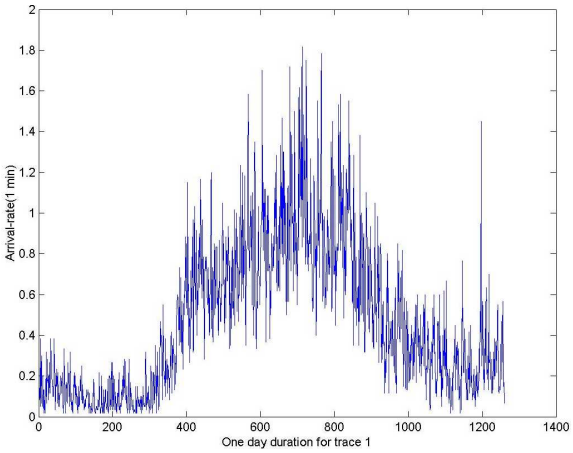

Figure 2: One day's arrival rates

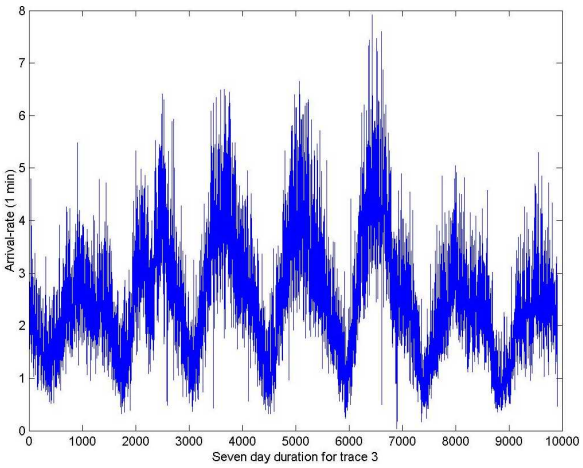

Figure 3: One week's arrival rates

Requests, in terms of volume, to web servers vary drastically through the day (see Figure 2 plotted from [8]) and through the week (see Figure 3 plotted from [7]). In order to efficiently operate web servers it is important to control traffic appropriately at different times of the day. Firstly it is required to determine how often to evaluate and change the price p. The obvious answer "whenever the arrival rates change significantly" is not easily implementable as the traffic is extremely stochastic. Therefore it is not clear if the traffic rate changes are purely statistical. In order to appropriately breakdown the data sets into intervals of time, this paper starts with 24 hours as the initial interval and reduce it until it is observed that the inter arrival times are iid. Although for extremely self-similar traffic such an interval may be hard to find, for the traces used here (specifically [8, 6, 7]) it was found that within a 1-hour interval all traces exhibited iid inter-arrival times.

The main criterion for deciding pricing and control strategy is revenue obtained by the web

service provider. Note that customer QoS (in terms of slowdown) is taken into account by charging a penalty. By autonomically setting price as well as admitting arriving requests, the revenue gained by the web server is maximized. The question to ask is what price in each period (of one hour) and control strategies to use. In order to determine this, at first the system workload has been studied, both in terms of arrival patterns and service requirements.

This paper assumes that workload information in terms of arrival rates are not available. It uses models based on time-series analysis to predict the mean arrival rates based on recent history as well as time-of-day (or seasonality effects). These time-series models are trained using historical data. Various models to predict workload have been considered (based on [18]) and the auto-regressive AR(1) model with seasonality SAR(2) is the final choice. Actual arrival rates versus predicted arrival rates for the three traces [8, 6, 7] are illustrated in Figure 4.

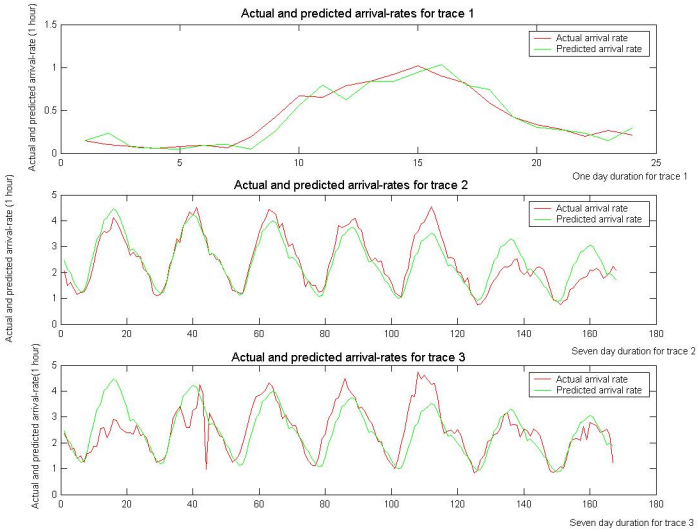

Figure 4: Actual versus predicted arrival rates

The time to serve (or process) a request is directly proportional to the size of the document requested. Using historical data the mean and variance of service times are obtained which are used in the analysis. For the analysis it is assumed that service times are independent and are sampled from an appropriate distribution (such as Pareto) whose parameters are obtained from the estimates of mean and variance of service times. However for the numerical calculations, real traces are used which include size of the document requested.

The optimal solution to the combined problem of pricing and admission control is intractable [23]. The paper develops three heuristics and compare them with a base line "do nothing" strategy. The next four subsections describe the following four strategies respectively: 1. Do nothing; 2. Static pricing and queue-length-threshold-based admission control; 3. Dynamic optimal pricing under no admission control; 4. Static pricing and nonnegative-profit-based admission control.

4.1 Strategy 1: Do nothing

This strategy is called "do nothing" because nothing is done to control admission of a request to the server. All requests are assigned price $p = 0$ in every interval. In addition, all arriving requests are admitted into the server. Therefore this strategy serves as a benchmark as it most closely represents the current web server implementation. By comparing other policies with this one, the improvement in revenue that one can expect by appropriately tuning the knobs, can be determined . Notice that users are still charged a price a per unit time served and a discount of c per unit of slowdown. This strategy clearly forms a lower bound on the revenue. For the fifth day of [7] the arrival rates are plotted along with the price and the admission policy (1 implies admit and 0 reject) in Figure 5.

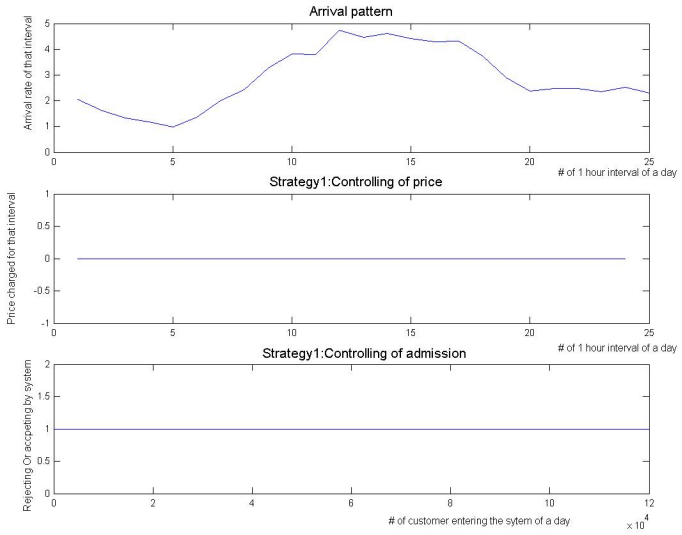

Figure 5: Arrival rates, price and admission policy for Strategy 1

4.2 Strategy 2: Queue length threshold and static price

Gautam and Seshadri [11] suggest that in web servers with self-similar arrivals, by selectively turning away requests in periods of high demand, the system performance improves dramatically. They show that by turning away about 5% of the requests at appropriate times, the queue lengths can be reduced by half. This paper takes advantage of this finding for the strategy described below. All users are charged a static price p, and customers are rejected whenever the number in the system exceeds L. Obtaining optimal values of p and L are intractable. It uses a trial-and-error technique to obtain the p and L values. For the fifth day of [7] the arrival rates have been plotted along with the price p and the admission policy under L (1 implies admit and 0 reject) in Figure 6. Although for Figure 6 and other numerical results, this paper obtains p and L by trying various values off-line by dumping the trace, while implementing, various p and L values should be tried on different days before settling on one.

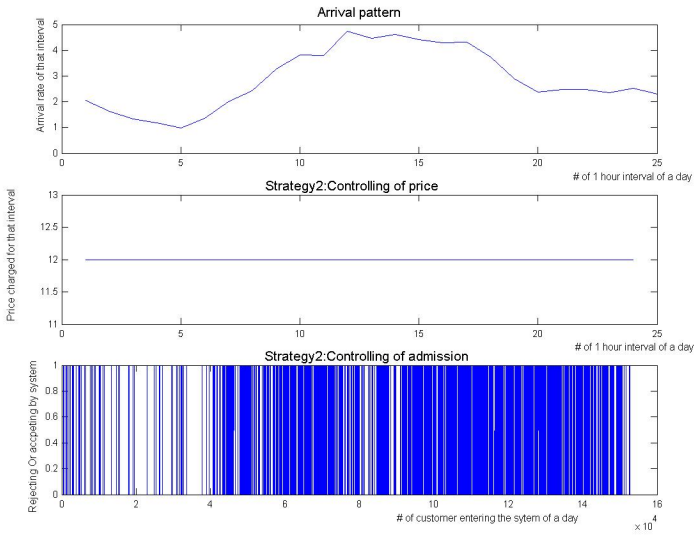

Figure 6: Arrival rates, price and admission policy for Strategy 2

4.3 Strategy 3: Optimal price under no drop

This strategy solves the optimal price p to charge in each interval (in this case, of 1 hour length) to maximize R in Equation (2), however no requests are rejected at the admission control stage. Notice that this scheme requires lesser information than Strategy 2 as it does not need to maintain any real-time system state information. In a given interval let $\overline{\lambda}$ be the estimated potential request arrival rate. The actual request arrival-rate λ is calculated, such that $q = \lambda/\overline{\lambda}$ is the fraction of traffic that accepts price p and enters the system. Based on the user-defined function $f(p)$ in Equation (1) between price p and acceptance probability q, it is known that given a p, $\lambda = f(p)\overline{\lambda}$. In order to keep the analysis tractable, it is assumed that in a period, the request arrivals are according to a Poisson process with mean rate λ requests per unit time. For an arbitrary arrival into the system, let W_q be the waiting time experienced in the queue (not including service) for this request. Although the response time W and the service time S are correlated, since $W_q = W - S$ is independent of S, the expected slowdown can be written as

$$E\Big[\frac{W}{S}\Big] = E\Big[1 + \frac{W_q}{S}\Big] = 1 + E[W_q]E\Big[\frac{1}{S}\Big]. \tag{3}$$

Inserting the inverse function of p in terms of λ, i.e., $p = f^{-1}(\lambda/\overline{\lambda})$ in Equation (2) and using the Pollaczek-Khintchine formula (assuming steady state is reached in the 1 hour interval) for W_q,

$$R = \lambda\left(f^{-1}(\frac{\lambda}{\overline{\lambda}})aE[S] - c(1 + E[W_q]E[1/S])\right)$$

$$= \lambda\left(f^{-1}(\frac{\lambda}{\overline{\lambda}}) + aE[S] - c(1 + \frac{0.5\lambda E[S^2]}{1 - \lambda E[S]}E[\frac{1}{S}])\right). \tag{4}$$

In order to obtain $E[1/S]$ in the above equation, the distribution of the service times needs to be known. If the service times are according to a Pareto random variable with cumulative distribution

function $\{1-(K/x)^{\beta}\}$, the revenue per unit time in terms of λ can be computed as

$$\begin{aligned} R &= \lambda\left(f^{-1}(\lambda/\overline{\lambda}) + a\frac{\beta K}{\beta-1} - c \right. \\ &- \left. \frac{\lambda c K \beta^2(\beta-1)}{2(\beta-2)(\beta+1)(\beta-1-K\beta\lambda)}\right). \end{aligned} \tag{5}$$

Taking the derivative of R w.r.t. λ in Equation (5) and setting it to zero, the optimal λ^* in the interval $[0, \min(\overline{\lambda}, 1/E[S])]$ is obtained. Thereby the optimal price p^* can be calculated in terms of λ^* by $p^* = f^{-1}(\lambda^*/\overline{\lambda})$. Notice that p^* is a function of $\overline{\lambda}$ and each 1-hour interval has a different potential arrival rate $\overline{\lambda}$, the price p^* is different. For the fifth day of [7] the arrival rates have been plotted along with the optimal price p^* and the admission policy (1 implies admit and 0 reject) in Figure 7.

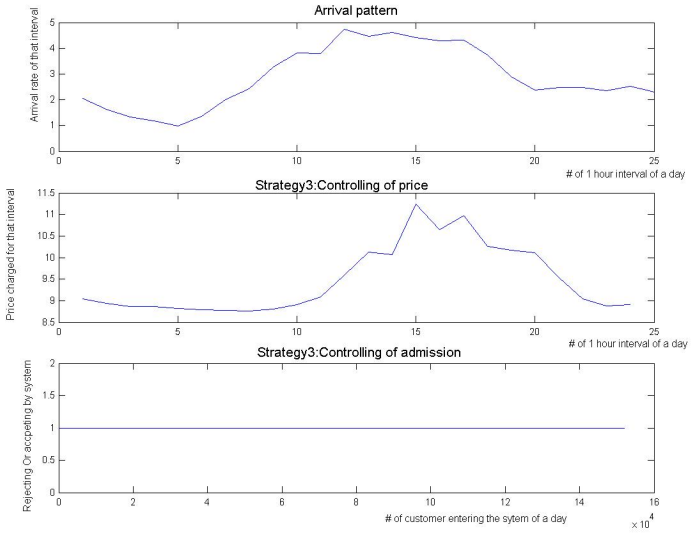

Figure 7: Arrival rates, price and admission policy for Strategy 3

4.4 Strategy 4: Non-negative profit and static price

This strategy drops requests whenever the revenue from that customer is negative. In achieving this, the server needs to maintain a lot of information such as total service time requirements of all requests ahead and including the new arrival. In addition, all users are charged a fixed and static price p. Obtaining optimal values of p in closed form is analytically intractable. However a trial-and-error technique has been used to obtain p.

For the fifth day of [7] the arrival rates have been plotted along with the price p and the above admission policy (1 implies admit i.e. when revenue is positive and 0 reject otherwise) in Figure 8. Similar to Strategy 2, for the results in Figure 8 and others in this paper, p has been obtained by trying various values on the same trace, however while implementing, various p values should be tried on different days before settling on one.

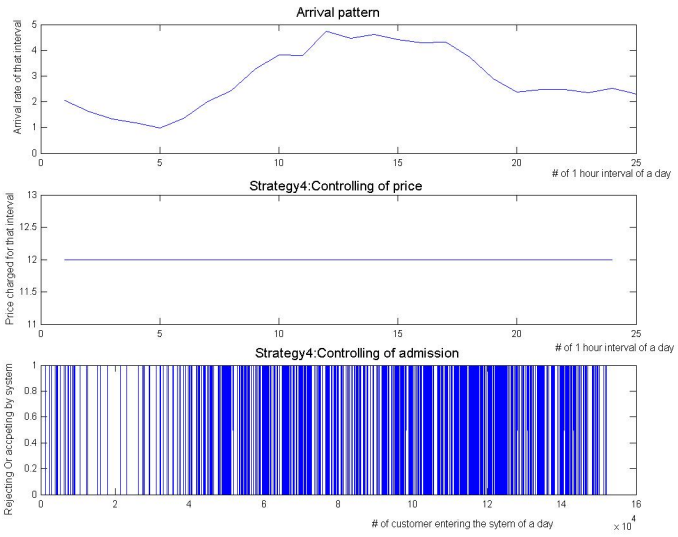

Figure 8: Arrival rates, price and admission policy for Strategy 4

5 Evaluation Results

The four strategies described in subsections 4.1 – 4.4 are recapitulated first. Consider a potential customer that arrives at time t. Upon knowing the price p at time t, the customer joins the system with probability $f(p)$ and leaves with probability $1 - f(p)$. If the customer were to enter the system at time t, the total number of customers in the system would be $X(t)$ and the total work remaining for the server to complete (i.e. response time of this customer) is $W(t)$. Thereby the revenue gained $R(t)$ upon accepting this customer (with service-time requirement $S(t)$) for service is $R(t) = p + aS(t) - cW(t)/S(t)$. The server needs to make 2 decisions: the price p and admission control (whether or not to admit the request after it enters the system). Table 1 summarizes he four strategies for these two decisions where information on implementability and computation complexity for each strategy is listed as well.

In order to compare the four strategies, numerical experiments have been performed and evaluation for the revenue obtained under the four strategies has been done. Eight experiments have been performed with different numerical values illustrated in Table 2. In that table, $\overline{\lambda}_{tr}$ denotes the mean arrival rate over an entire 7 day period. For all the experiments the pricing function is $f(p) = 1 - p^2/G^2$. For tests 1 to 4, trace [7] was used and for tests 5 to 8, trace [6] was used.

Upon running multiple replications of simulations on the real traces of the eight test problems, grand average revenue per second for the four strategies was obtained. The results are illustrated in Figure 9. It is clear from Figure 9 that the do nothing strategy performs very poorly. Hence it can be concluded that by rejecting some customers (either via a pricing scheme or admission control) it is possible to make the system extremely profitable. In addition, it must be noted that in all cases strategy 3 which uses very minimal information performs reasonably well. In fact to obtain p for strategies 2 and 4 is extremely time-consuming and has to be done by trial and error. In addition, strategy 3 is fair to customers of various sizes in terms of dropping customers. In order to avoid negative revenue, strategy 4 will reject small file size requests. Strategy 2 is not as bad as strategy 4 but it serves much fewer smaller sized files than strategy 3.

	p	*Admission policy*
1	$p = 0$ always; Easy to implement; Computation easy.	Always admit; Easy to implement; Computation easy.
2	Computed by search; Does not vary from interval to interval; Easy to implement; Computation hard.	Admit if $X(t) < L$ where L is found by trial and error; need $X(t)$ to implement; Computation hard.
3	$p = argmax\{R\}$ using formula; need arrival rate and service time moments; p varies from interval to interval; Update p to implement; Computation easy.	Always admit Easy to implement; Computation easy.
4	Computed by search; Does not vary from interval to interval; Easy to implement; Computation hard.	Admit if $R(t) > 0$ need $R(t)$ to implement; Computation easy.

Table 1: Comparing the 4 strategies

In terms of comparing revenues for the different strategies, see Figures 10, 11 and 12. There are no graphs for strategy 1 as it does not incur any positive revenue. There are three sources of revenue, the fixed revenue (p term), the revenue based on file sizes (a term) and the cost for reimbursing customers experiencing huge slowdowns (c term). For most strategies, when the revenue is low, medium and high, the c terms, p term and a term dominate the revenue respectively. Also notice that only a negligible fraction of requests have very high revenue or negative revenue (as compared to strategy 1). This indicates that all strategies in some sense perform very well and the spread of revenue is fairly homogeneous.

6 Concluding Remarks

This paper considers a web hosting service that charges users for the files they request. In return the web service provides QoS by offering discounts on the price if they experience slowdowns. Request arrival rates to web servers vary greatly during the course of a day (and a week). The web servers have two parameters they control - (1) the price a customer is charged which is based on arrival rates and the customer has the option of accepting or rejecting the service upon seeing the price; (2) if a customer accepts a price, the web server can accept or deny the request which is based on the state of the server in terms of number of jobs and amount of service remaining.

In order to make the two decisions (price and admission control) autonomically, this paper develops and evaluates 3 heuristics (with varying degrees of performance, implementability and computational complexity) to solve the optimization problem at different points in time. This is

Test	a	c	G	$\overline{\lambda}_{tr}$	$E(S)$	$\sqrt{Var(S)}$
1	40	0.04	20	2.49	0.251	0.987
2	40	0.04	15	2.49	0.251	0.987
3	4	0.04	20	2.49	0.251	0.987
4	4	0.04	15	2.49	0.251	0.987
5	4	0.07	20	2.42	0.166	0.62
6	4	0.07	15	2.42	0.166	0.62
7	40	0.07	20	2.42	0.166	0.62
8	40	0.07	15	2.42	0.166	0.62

Table 2: Numerical values for experiments

Figure 9: Revenue per second using the 4 strategies

with the understanding that it is intractable to make both decisions in an optimal and dynamic manner. The three heuristics are compared against a 4^{th} strategy "do nothing" policy that accepts all customers and charges them with p = 0 . It is observed that the "do nothing" policy performs very poorly. However the policy where only the price is controlled but all entering customers are accepted, performs reasonably well. This policy also uses very little information about the state and the environment, and is fair. Other policies such as the static threshold policy (strategy 2) and, non-negative profit and static price policy (strategy 4) are also evaluated. They perform extremely well, indicating that an appropriately chosen static price with admission control can perform similar or better than an adaptive price scheme with no admission control.

Using the results of this initial study, the authors intend to extend the analytical model (strategy 3) to incorporate admission control as it promises to improve the performance. In addition they will investigate other models for obtaining the price, although they must say that the Poisson approximation for traffic did perform reasonably well. In future the authors would like to extend the analysis to other scheduling policies besides FCFS. The user behavior with respect to other price parameters such as a and c has also been considered . Other extensions include multiple classes of traffic, power issues, etc.

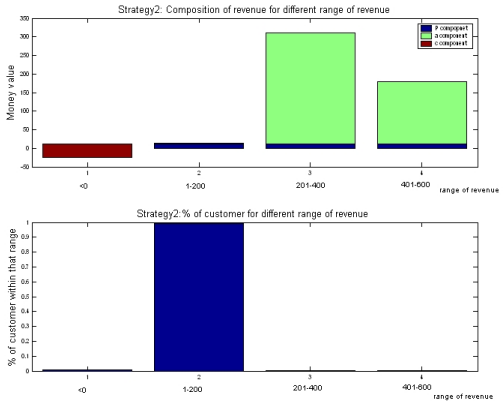

Figure 10: Revenue distribution for strategy 2

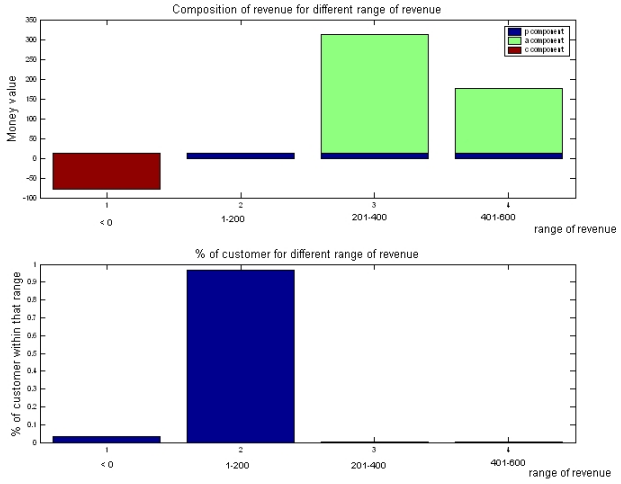

Figure 11: Revenue distribution for strategy 3

Acknowledgements

This research is partially supported by NSF grant ACI-0325056 “Data-driven Autonomic Performance Modulation for Servers”.

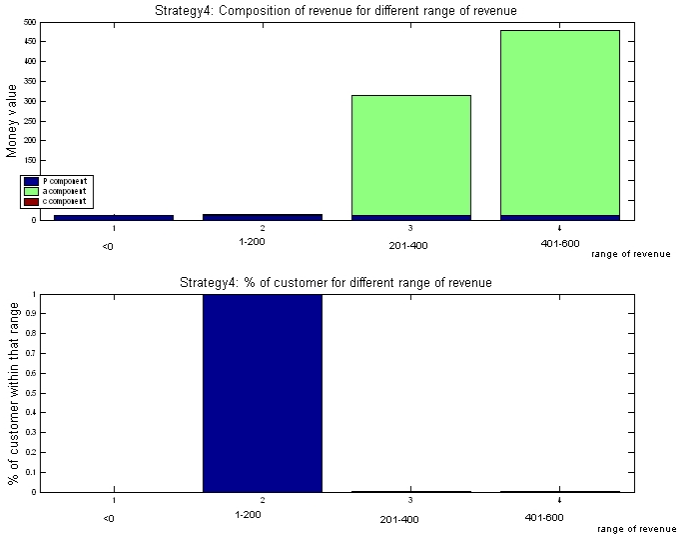

Figure 12: Revenue distribution for strategy 4

References

[1] S. H. Clearwater, 1996. Market-based control. A paradigm for distributed resource allocation. World Scientific, Singapore.

[2] B. A. Huberman, 1988. The Ecology of Computation. North-Holland, Amsterdam.

[3] Buyya. R, 2002. Economic-based Distributed Resource Management and Scheduling for Grid Computing. Monash University, Melbourne, Australia.

[4] T. F. Abdelzaher and C. Lu, 2000. Modeling and performance control of Internet servers. In *39th IEEE Conference on Decision and Control, Sydney, Australia*, pages 2234–2239.

[5] E. Anderson, M. Hobbs, K. Keeton, S. Spence, M. Uysal, and A. Veitch, 2002. Hippodrome: running circles around administration. In *FAST '02, Monterey, CA*, pages 175–188.

[6] S. Balbach, 1995. Clarknet-http server logs, August 28, 1995 - September 03, 1995. In *http://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html*.

[7] S. Balbach, 1995. Clarknet-http server logs, September 04, 1995 - September 10, 1995. In *http://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html*.

[8] L. Bottomley, 1995. Epa-http server logs. In *http://ita.ee.lbl.gov/html/contrib/EPA-HTTP.html*.

[9] A. Chandra, W. Gong, and P. Shenoy, 2003. Dynamic resource allocation for shared data centers using online measurements. In *ACM/IEEE International Workshop on Quality of Service (IWQoS), Monterey, CA*, pages 381–400.

[10] B. Chun and D. Culler, 2002. User-centric performance analysis of market-based cluster batch schedulers. In *2nd IEEE Intl. Symp. on Cluster Computing and the Grid, Berlin, Germany*, pages 22–30.

[11] N. Gautam and S. Seshadri, 2002. Performance analysis for e-business: Impact of long range dependence. *Electronic Commerce Research*, 2(3):233–253.

[12] IBM. Total cost of ownership, 2003. In *http://www-1.ibm.com/servers/solutions/serverconsolidation/tco/*.

[13] C. Lu, T. Abdelzaher, J. Stankovic, and S. Son, 2001. A feedback control approach for guaranteeing relative delays in web servers. In *IEEE Real-Time Technology and Applications Symposium, Taipei, Taiwan*, pages 51–62.

[14] I. Paschalidis and Y. Liu, 2002. Pricing in multiservice loss networks: Static pricing, asymptotic optimality, and demand substitution effects. *IEEE/ACM Transactions on Networking*, 10(3):425–438.

[15] B. Schroeder and M. Harchol-Balter, 2003. Web servers under overload: How scheduling can help. To appear in *18th International Teletraffic Congress, Berlin, Germany*.

[16] H. P. Services, 2003. Getting serious about tco. In *http://www.hp.com/hps/spotlight/index_tco.html*.

[17] K. Shen, H. Tang, T. Yang, and L. Chu, 2002. Integrated resource management for cluster-based Internet services. In *Fifth USENIX Symposium on Operating Systems Design and Implementation (OSDI), Boston, MA*, pages 225–238.

[18] R. Shumway and S. Stoffer, 2000. *Time Series Analysis and its Applications*. Springer, New York, NY.

[19] V. Sundaram and P. Shenoy, 2003. A practical learning-based approach for dynamic storage allocation. In *ACM/IEEE International Workshop on Quality of Service (IWQoS), Monterey, CA*, pages 479–497.

[20] B. Urgaonkar, P. Shenoy, and T. Roscoe, 2002. Resource overbooking and application profiling in shared hosting platforms. In *Fifth Symposium on Operating Systems Design and Implementation (OSDI), Boston, MA*, pages 239–254.

[21] A. Wierman and M. Harchol-Balter, 2003. Classifying scheduling policies with respect to unfairness in an m/g/1 queue. In *ACM SIGMETRICS, San diego, CA*, pages 238–249.

[22] Q. Zhang, E. Smirni, and G. Ciardo, 2003. Profit-driven service differentiation in transient environment. In *MASCOTS, Orlando, FL*, pages 230–233.

[23] S. Stidham, 1992. Pricing and capacity decisions for a service facility: stability and multiple local optima. In *Management Science*, 38(8): pages 1121–1139.