Pricing Issues in Web Hosting Services

Natarajan Gautam*
Harold and Inge Marcus Department of Industrial and Manufacturing Engineering
The Pennsylvania State University
310 Leonhard Bldg., University Park, PA 16802
Email: ngautam@psu.edu
Phone: 814-865-1239
Fax: 814-863-4745

Initial Submission: March 20, 2004
Revised Submission: September 25, 2004

*__Short description of the author__: Natarajan Gautam is an Associate Professor of Industrial Engineering at Penn State. He received his B.Tech. degree in Mechanical Engineering from the Indian Institute of Technology, Madras, and his M.S. and Ph.D. in Operations Research from the University of North Carolina at Chapel Hill. He is a member of INFORMS, IIE and IEEE. He is an Associate Editor for the INFORMS Journal on Computing. He served as the Newsletter Editor as well as Website Editor for the INFORMS Applied Probability Society.

Pricing Issues in Web Hosting Services

Abstract

In this paper we consider a web hosting application where a web server hosts a website for a client organization. Users that access this web server are sensitive to the quality of service (QoS) they receive (such as response time and request loss). In the current state of practice, web hosting services charge clients only based on the amount of information stored but do not negotiate any quality of service details. This is changing as users expect good QoS. From a cost standpoint, these web server farms consume a lot of electric power which needs to be incorporated in the price charged to clients. In addition, the Internet domain in the web server farm vicinity experiences severe congestion. In this paper we consider two main parameters that the web servers tune, namely, caching strategy which is a one time decision and the processing speed to run the web server which can be tuned from time to time. We present a detailed methodology that web servers can incorporate to make the above decisions. Based on the dynamics between the three players (users, clients and web servers), we show that the price and web server settings could both either stabilize or oscillate between several points indicating chaos.

Keywords: pricing, quality of service, queueing model, fixed point iteration

1 Introduction

In this day and age, every company or firm needs to maintain its presence in the Internet through their web sites. However, it is not possible for all organizations to have the workforce, skill, and money to set up and maintain a web server locally. Therefore these organizations outsource their web sites to companies that specialize in web hosting. These web hosting companies own very large web server farms consisting of thousands of servers where they store web sites of their clients. The current state of practice in terms of pricing schemes for web hosting services, is to charge a fixed price for the amount of information (in bytes) stored in the web server. Organizations such as Dotster (2004), Fortune-City (2004), Global-Servers (2004) and most other web hosting companies use such a pricing scheme.

The problem with the current pricing scheme is that there is no mention of quality of service (QoS) that the users experience. All Internet applications are rapidly incorporating QoS guarantees, hence users are very sensitive to the lack of QoS. Since the number of users and applications in the Internet is growing at an exponential rate, the clients would soon start demanding QoS for their users in order to stay competitive. However, what is complicated is that clients do not directly interact with their customers, hence the clients would have to periodically monitor the web servers to determine if the users' QoS is met. Few companies such as Website-Providers (2004) promise a 99.5% up time guarantee (as a QoS measure) and also require the clients to adhere to certain traffic requirements. However, this does not help the user much as the QoS could degrade due to congestion in the Internet domain in the vicinity of the web server farm. The ultimate objective is to provide guarantees that can be directly related to the performance experienced such as response time and loss.

The objective of this paper is twofold. Firstly, it provides a tool for performance analysis and decision making for web servers. In particular, we recommend strategies for caching policies as well as for determining performance measures, such as response time and loss. The second objective is to study a pricing model for web hosting, that web servers can use iteratively by periodically measuring user characteristics and updating prices with the client. The paper is written with the web server farm as a potential beneficiary, however we incorporate client and user behavior aspects in such a manner the clients could also benefit in terms of being able to negotiate an appropriate contract. In principle, this paper studies the interaction between the three parties: users, clients

and web servers.

The literature on pricing issues in the Internet and telecommunication networks has gained importance in the last two decades. We first summarize some of the work in the previous decade. Kelly (1996) describes a charging and accounting mechanism based on an effective bandwidth concept. Edell, McKeown, and Varaiya (1995) present a system for billing users for their TCP traffic. MacKie-Mason and Varian (1995) discusses issues on pricing the internet. Parris, Keshav, and Ferrari (1992) present a framework for pricing services, and study the effect of pricing on user behavior and network performance. Cocchi, Estrin, Shenker, and Zhang (1993) study the role of pricing policies in multiple service class networks. Crowcroft (1996) proposes subscription based pricing, and a mechanism based on dynamic host address allocation giving priority levels for usage. Shenker, Clark, Estrin, and Herzog (1996) state that the research agenda on pricing in computer networks should shift away from the optimality paradigm and focus on structural/architectural issues. Some of the other work, especially analytical-model based results, are summarized in the recent book by Courcoubetis and Weber (2003).

In the current decade a large number of research articles have appeared on pricing in the Internet. Caesar, Balaraman, and Ghosal (2000) present a comparative study of usage-based pricing strategies for telephone services over the Internet. Adler, Cai, Shapiro, and Towsley (2003) consider algorithms for probabilistically setting explicit congestion notification bit at routers to estimate link congestion prices. Odlyzko (2000) argues that although flat rate continues to be the predominant form of selling Internet access, flat rate pricing encourages waste and requires the 20 percent of users who account for 80 percent of the traffic to be subsidized by other users and other forms of revenue. In (Davies, Hardt, and Kelly 2004), the authors describe a framework for capacity planning and costing in an IP network environment. In a congestion-pricing framework, according to Henderson, Crowcroft, and Bhatti (2001), the congestion charge would replace usage and QoS charges. Thereby, users would pay their Internet service providers a subscription charge to cover fixed costs and a congestion charge only when appropriate. Flat rate and user based pricing policies are used in (Hamadeh, Jin, Walia, Kesidis, and Kirjner 2004) to consider issues of cost recovery in residential broadband access. Turner and Ross (2004) consider a market where peers trade resources such as storage, bandwidth and CPU cycles. The authors build a protocol to facilitate this. Jin and Jordan (2004) investigate the sensitivity of resource allocation and the resulting QoS to resource prices in a reservation-based QoS architecture that provides guaranteed bounds on packet

loss and end-to-end delay for real-time applications.

The paper is organized as follows. In Section 2, we describe the problem scenario in order to model the system to predict performance. In Section 3, we describe a methodology to make the one-time decision of choosing an appropriate caching mechanism. The other performance analysis, namely the queueing model to determine the QoS measures such as response time and loss, is described in Section 4. Once the performance models are sorted out, we turn to pricing aspects and study the relationship between the three groups: users, clients and servers, in Section 5. Finally in Section 6 we state our concluding remarks and point out some future directions of work.

2 System Performance Modeling

There are three players in this system: web servers, client and users. Typically a web-hosting service will own thousands of web servers (called server farms). Each web server hosts web sites for one or more clients. Users access the client's web pages from the web server and experience certain QoS. Typically when users of a client have stringent QoS requirements, the client usually gets a dedicated web server. If the web server hosts more than one client, typically they either have similar QoS needs and workload characteristics, or they have no QoS requirements. In the former case, we can club all the clients' users and think of the web server as though it caters to a single client. In that light, we can model each web server independent of the other web servers in the farm, with the understanding that for all practical purposes it is as though there is a single client. The only time a web server feels the effect of other web servers is when there is a congestion caused in the local network (called domain) of the web server farm.

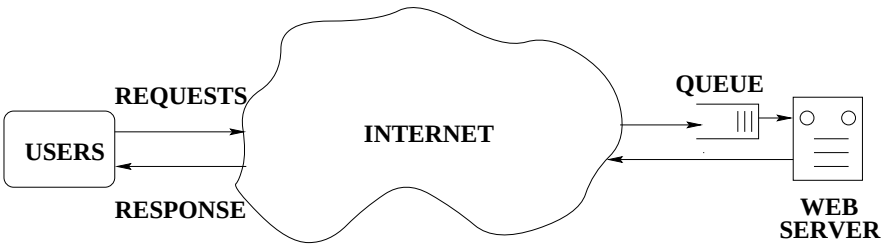

Figure 1: The system of users, a single web server and the Internet

Consider a system of users, a single web server, and the Internet that connects the users to the web server. See Figure 1 for an illustration of the system. Assume that the user requests come according to a Poisson process with parameter λ requests per minute. This assumption is made for

analytical convenience with the understanding that in reality not only does λ not stay a constant, but the inter-arrival times of requests some times are non-exponential. The requests to the web server wait in a queue in front of the server. The web server serves one request at a time according to first-come-first-served policy. We assume that all requests are for files that can be transmitted to the user as a string of packets, none of the requests are for audio or streaming video applications that have bandwidth requirements. The only two QoS requirements we consider in this paper are: delay (average time between request and response) for those requests that receive a response and request loss (fraction of requests that were lost without a response).

2.1 Modeling Service Times

The web server stores documents in two places: one in its main memory which takes a while to retrieve and the other at a cache which can be retrieved relatively quickly. Typically only a small number of documents are stored in the cache. When a web server picks up a request from its queue, it first checks the cache to see if the request can be satisfied (if yes, the response is given quickly and is called *cache hit*), otherwise the request is satisfied from the main memory (this is called *cache miss*). This is pictorially explained in Figure 2.

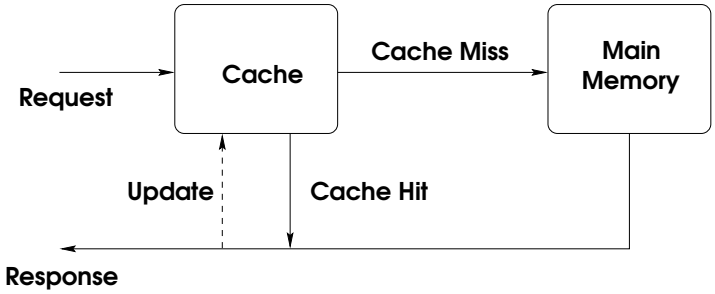

Figure 2: The web server cache and main memory

The web server can update the contents of the cache based on the information requested. In addition, the cache can be sorted in an appropriate manner to minimize searching speed. There are several strategies available in the literature for cache management. We will address these caching schemes in Section 3. In particular we will obtain the cache miss probability (called cm) and cache hit probability. Let the average time to locate the requested file in the cache be ℓ_c. Also let the average time to locate a requested file in the main memory (including time spent searching the cache) be ℓ_m. If the average file size is M bytes and processor speed is c bytes per minute, then

the average service time (in minutes) for a request is

$$S = M/c + cm(\ell_m) + (1 - cm)(\ell_c). \tag{1}$$

2.2 Modeling Request Loss

A request is considered lost when either the user's request cannot reach the web server, or the web server's response cannot reach the user. Both happen when there is congestion in the web server farm local network (or domain) or the web server itself goes down. We assume that the web server down times as well as network congestion durations are long enough that the requests waiting in the web server queue can be dropped. In addition any requests that come during this down (or congested) period are also lost. Thereby request loss is computed as: the sum of fraction of requests that were dropped and fraction of requests that were lost.

We can model this on-off behavior using an alternating renewal process. When the web server is up and the domain network is congestion free, it corresponds to the server being "on". Likewise, if either the web server is down, or the domain network is congested or both, it corresponds to the server being "off". Thereby we can model the system as a single server queue with Poisson arrivals and infinite waiting space such that when the server is on, it serves at an average rate of $1/S$ (see Equation (1)) customers per minute. When the server goes from on to off, all requests in the queue and server are dropped. The system stays empty until the server comes back on. For analytical convenience, we assume that the service times are exponentially distributed with mean $\frac{1}{\mu} = S$. Also the on times and off times are assumed to be exponentially distributed with mean $\frac{1}{\alpha}$ and $\frac{1}{\beta}$ minutes respectively. This system is analyzed in Section 4. Although several intricate queueing network models are used (Menasce and Almeida 1998) to model web servers, they do not incorporate congestion issues. Here we model the web server as a single station with the understanding that only the bottleneck node is modeled and only catastrophic server breakdowns are modeled.

2.3 Practical Considerations

Before jumping into the analytical models, it may be worthwhile to state the appropriateness of some of the assumptions made thus far, especially from a practical point of view. We have assumed for our model that the inter-arrival times, service times, on times and off times are all exponentially distributed. Traces from web servers (Liu, Niclausse, and Jalpa-Villanueva 2001) have shown that Poisson process is usually sufficient to model session arrival process. Since Poisson

process implies exponentially distributed inter-arrival times, this assumption is valid. However, service times are not usually exponential in practice. This is a first-order approximation made for modeling convenience. With regards to on and off times, Gray and Siewiorek (1991) reports that practical measurements reveal that off-times are exponentially distributed, whereas on-times are usually obey a Weibull distribution or a negative hyper-exponential distribution. Other assumptions made henceforth would be supported by references.

3 Evaluating Caching Schemes

We consider two main caching strategies (Mookerjee and Tan 2002) in web proxy servers, LRU (least recently used) and LFU (least frequently used). We obtain analytical expressions for the cache-miss-probability for both LRU and LFU based on a given user behavior pattern. In the literature there are several papers that develop approximations for analyzing these caching strategies, however there is very little exact analysis, as done in this paper. There are two Operations-Research papers that devote a large effort analyzing caching strategies, namely (Mookerjee and Tan 2002 ; Jelenkovic and Radovanovic 2004). In particular, Mookerjee and Tan (2002) build a deterministic model that accurately computes cache properties based on sampled data. Further, Jelenkovic and Radovanovic (2004) study a stochastic model of a cache, and obtain optimal control decisions in a dynamic environment. This paper considers the caching problem as a strategic problem (as opposed to operational problems as considered by Mookerjee and Tan (2002) as well as Jelenkovic and Radovanovic (2004)).

We first explain the notations and the caching strategies. Then we model the LFU scheme (Section 3.1) followed by the LRU scheme (Section 3.2). Finally we illustrate the results using an example in Section 3.3. Consider a single web server. Let $\mathcal{B}$ be the set of all possible information that the users access this web server for. Let B be the cardinality of $\mathcal{B}$, i.e. B is the total number of all possible information that can be cached. At any given time in this web server, a subset of $\mathcal{B}$ is cached. Let the cache size be H. That means at a time, a maximum of H (where $H \leq B$) pieces of information can be cached in this web server.

When a user requests for a piece of information and this is not available in the cache, we encounter what is known as a cache-miss. Our main objective is to obtain analytical expressions for this cache-miss-probability under two caching strategies, LRU and LFU. In the LRU scheme,

the cache is arranged according to an increasing least-recently-used scheme. That means the most recently used information is at the head of the line in the cache queue. However, in the LFU scheme, the last T requests are considered and the cache entries are arranged in an increasing least-frequently used order. That means the most frequently accessed information among the last T requests, is at the head of the line in the cache queue.

We assume that all users are identical and choose information u with probability $p(u)$ (where $u \in \mathcal{B}$). The probability mass function of the chosen information is depicted in Figure 3. We use a static model where this probability mass function of the chosen information does not change with time. However, in the dynamic case we can assume quasi-static behavior and use this model. Based on several papers (such as Breslau, Cao, Fan, Phillips, and Shenker (1999)) we model the probability mass function using Zipf distribution for our numerical computations.

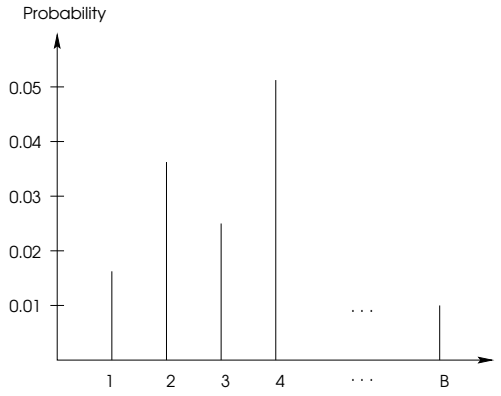

Figure 3: Probability mass function of the chosen information

3.1 The LFU scheme

In this scheme, the index corresponding to the last T requests are stored. Based on the frequency of the indices, in the long run the cache contains H (or less) entries in a decreasing LFU order. Let v_1, v_2, ..., v_T be the last T requests that are stored. Let $N_n(u)$ be the number of entries among the T entries that correspond to information u just before the n^{th} request. For $n > T$, let X_n^i be the index of the i^{th} entry of the cache just before the n^{th} request arrives. Note that $X_n^i = 0$, if i^{th} entry of the cache is empty. Therefore X_n^1 corresponds to the highest $N_n(u)$, X_n^2 corresponds to the next highest $N_n(u)$, and so on. The vector $(X_n^1, X_n^2, \ldots, X_n^H)$ represents the cache in an increasing LFU order just before the n^{th} request arrives. The dynamics of the $N_n(\cdot)$ process is described as follows: if u is the index corresponding to the n^{th} user request (this happens with probability $p(u)$)

then $N_{n+1}(u) = N_n(u) + 1 - I_{v_1=u}$, where $I_{v_1=u}$ is the indicator variable that takes on the value 1 if $v_1 = u$, and 0 if $v_1 \neq u$. Also, for $i \neq u$,

$$N_{n+1}(i) = \begin{cases} N_n(i) - 1 & \text{if } i = v_1, \\ N_n(i) & \text{if } i \neq v_1. \end{cases}$$

Let $U = (u_1, u_2, \ldots, u_B)$ be the ordered set corresponding to the decreasing order of the probability mass function, i.e., u_i and u_j are such that if $p(u_i) > p(u_j)$ then $i < j$. The following theorem describes the cache miss probability in the long run based on the limiting distribution of the cache contents.

Theorem 1 *For large values of T (specifically as $T \to \infty$), in the limit X_n^j (for $j = 1, \ldots, H$) can be calculated as*

$$\lim_{n\to\infty} X_n^j \to u_j \quad \textit{with probability 1.}$$

Hence the long-run probability of cache miss under LFU is

$$cm_{lfu} = (1 - u_1 - u_2 - \ldots - u_H).$$

Proof. The theorem is a direct consequence of the strong law of large numbers (SLLN). By making suitable transformation of variables, particularly Bernoulli random variables, and then using SLLN one can prove the above result. ■

3.2 The LRU scheme

In this scheme, we assume that in steady-state, the cache contains H entries in an increasing LRU order. Let Y_n^i be the index of the i^{th} entry of the cache just before the n^{th} request arrives. The vector $(Y_n^1, Y_n^2, \ldots, Y_n^H)$ represents the cache in an increasing LRU order just before the n^{th} request arrives. The dynamics of the cache is described as follows: if u is the index corresponding to the n^{th} user request (this happens with probability $p(u)$), then check if $u \in \{Y_n^1, Y_n^2, \ldots, Y_n^H\}$. If indeed $u \in \{Y_n^1, Y_n^2, \ldots, Y_n^H\}$, in particular, $u = Y_n^j$, then the vector $(Y_{n+1}^1, Y_{n+1}^2, \ldots, Y_{n+1}^H)$ is such that

$$Y_{n+1}^i = \begin{cases} Y_n^j & \text{if } i = 1 \\ Y_n^{i-1} & \text{if } 1 < i \leq j \\ Y_n^i & \text{if } i > j. \end{cases}$$

However, if $u \notin \{Y_n^1, Y_n^2, \ldots, Y_n^H\}$, then the vector $(Y_{n+1}^1, Y_{n+1}^2, \ldots, Y_{n+1}^H)$ is such that

$$Y_{n+1}^i = \begin{cases} u & \text{if } i = 1 \\ Y_n^{i-1} & \text{if } i > 1. \end{cases}$$

Let $\mathcal{S}$ be the state space of the stochastic process $\{(Y_n^1, Y_n^2, \ldots, Y_n^H), n \geq 0\}$. Clearly, $\mathcal{S}$ can be written as

$$\mathcal{S} = \{(y_1, y_2, \ldots, y_H) : \forall i, j \in (1, 2, \ldots, H), \text{ if } i \neq j, \text{ then } y_i \neq y_j\}.$$

That means for a given n, no two elements in the H-tuple $(Y_n^1, Y_n^2, \ldots, Y_n^H)$ are identical. The following theorem describes the cache miss probability in the long run based on the limiting distribution of the cache contents.

Theorem 2 *The limiting distribution of the stochastic process* $\{(Y_n^1, Y_n^2, \ldots, Y_n^H), n \geq 0\}$ *is given by*

$$\begin{aligned}
\pi_{y_1,y_2,\ldots,y_H} &= \lim_{n\to\infty} P\{Y_n^1 = y_1, Y_n^2 = y_2, \ldots, Y_n^H = y_H\} \\
&= \frac{p(y_1)p(y_2)\ldots p(y_H)}{[1-p(y_1)][1-p(y_1)-p(y_2)]\ldots[1-p(y_1)-p(y_2)-p(y_3)-\ldots-p(y_{H-1})]}.
\end{aligned}$$

Hence the long-run probability of cache miss under LRU is

$$\begin{aligned}
cm_{lru} &= \sum_{(y_1,y_2,\ldots,y_H)\in S} \pi_{y_1,y_2,\ldots,y_H}(1-p(y_1)-p(y_2)-\ldots-p(y_H)) \\
&= \sum_{(y_1,y_2,\ldots,y_H)\in S} \frac{p(y_1)p(y_2)\ldots p(y_H)(1-p(y_1)-p(y_2)-\ldots-p(y_H))}{[1-p(y_1)][1-p(y_1)-p(y_2)]\ldots[1-p(y_1)-p(y_2)-p(y_3)-\ldots-p(y_{H-1})]}.
\end{aligned}$$

Proof. Writing down the balance equation for the discrete time Markov chain $\{(Y_n^1, Y_n^2, \ldots, Y_n^H), n \geq 0\}$, and going through the algebra, it is possible to show the results of the theorem. ■

3.3 Example

As a simplified example (to illustrate the results), consider there are 10 pieces of information (i.e. $B = 10$) and the cache size is 4 (i.e. $H = 4$). The following is the probability mass function of the 10 pieces of information:

$$\begin{array}{llllll}
p(1) = 0.11, & p(2) = 0.09, & p(3) = 0.07, & p(4) = 0.05, & p(5) = 0.17, \\
p(6) = 0.15, & p(7) = 0.03, & p(8) = 0.01, & p(9) = 0.13, & p(10) = 0.19.
\end{array}$$

3.3.1 LFU results

From the above values, it is clear that $U = (10, 5, 6, 9, 1, 2, 3, 4, 7, 8)$. Therefore if T is large, the cache in the long run would be $(10, 5, 6, 9)$ and the cache miss probability cm_{lfu} from Theorem 1 is $1 - p(10) - p(5) - p(6) - p(9) = 0.36$. The results match with the simulations. However on running simulations for smaller T values, we observe (based on a single simulation run for each T value) that as T increases, cm_{lfu} decreases asymptotically to 0.36. Figure 4 illustrates the results.

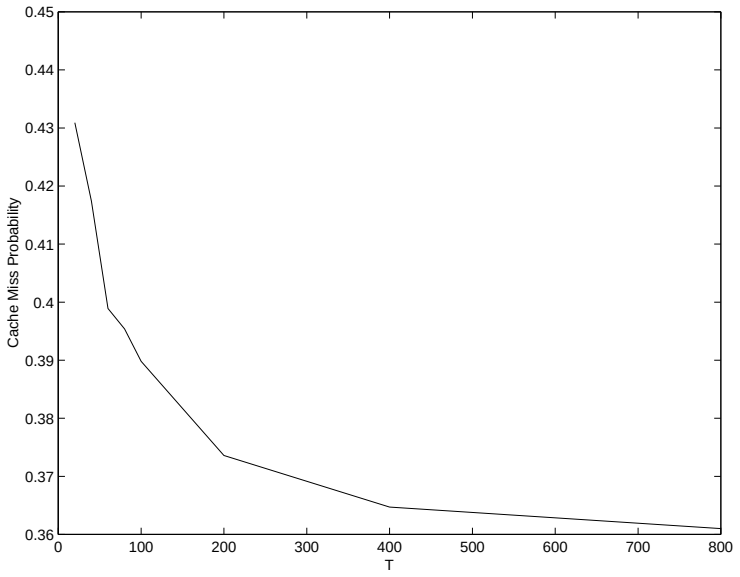

Figure 4: Probability of cache miss as a function of T

3.3.2 LRU results

From Theorem 2, the long run probability that the cache has entries $(5, 6, 9, 10)$ is given by

$$\begin{aligned}\pi_{5,6,9,10} &= \frac{p(5)p(6)p(9)p(10)}{[1-p(5)][1-p(5)-p(6)][1-p(5)-p(6)-p(9)]}. \\ &= 0.002.\end{aligned}$$

The above results agree perfectly with the simulations.

Likewise, the cache miss probability using Theorem 2 can be calculated as

$$cm_{lru} = \sum_{i=1}^{10} \sum_{\substack{j=1 \\ j\neq i}}^{10} \sum_{\substack{k=1 \\ k\neq i,j}}^{10} \sum_{\substack{l=1 \\ l\neq i,j,k}}^{10} \frac{p(i)p(j)p(k)p(l)[1-p(i)-p(j)-p(k)-p(l)]}{[1-p(i)][1-p(i)-p(j)][1-p(i)-p(j)-p(k)]} = 0.4879.$$

The above cm_{lru} value tallied well with the simulations, and thereby validating our theorem.

3.4 Implementation

We now discuss some implementation issues that web servers will face while determining an appropriate caching strategy. When the relative popularity of the web documents change fast, LRU is a better policy (since it does not retain old information). However if the popularity remains fairly consistent over time, LFU is better. It is important to note that the number of documents would be more-or-less fixed and hence due to the use of Zipf distribution, although the sorted ranks change, their individual probabilities relative to their popularity ranks remain the same. Hence the cache

hit and cache miss probabilities do not change significantly with time. Thus these results are used in the initial set up phase to aid a strategic decision. Once the caching strategy is chosen, the web server only concentrates on dynamically modifying server speed from period to period.

4 Single Server Queue with Breakdown and Repair

In this section, we continue the analysis from Section 2.2. Recall that we have a single server queue where customers arrive according to a Poisson process with mean arrival rate λ. Service times are exponentially distributed with mean $1/\mu$. There is infinite room for requests to wait. The server stays on for a random time distributed exponentially with mean $1/\alpha$. When the server turns off, all customers in the system are ejected. Then the server stays off for a random time distributed exponentially with mean $1/\beta$. No requests can enter when the server is off. Recall that the server is considered "off" either when there is congestion in the local network of the web server farm, or when the server actually goes down. Notice that the system behaves as an $M/M/1$ queue when the server is on, and the system is empty when the server is off. The server toggles between on and off states.

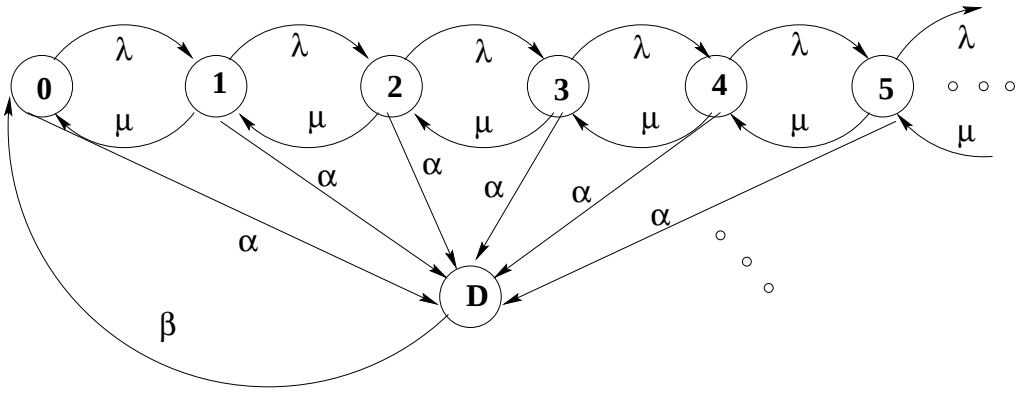

Figure 5: Rate diagram of the CTMC

Chao (1995) considers a similar problem where the breakdowns are called catastrophes as all the customers are lost. However the assumptions and settings in Chao (1995) are different in the following ways: (1) catastrophe occurs with arrivals in Chao (1995), however in this paper, the server randomly toggles between up and down states; (2) repair times are zero in Chao (1995); (3) the methodology in the two papers are different. However we agree with Chao (1995) that this type of catastrophic breakdowns have received very little attention. There have been some recent papers (for example Chao and Zheng (2003), Kumar and Arivudainambi (2000), etc.) that consider

transient analysis of such systems.

We model the system as a continuous time Markov chain (CTMC). Let $X(t) = i$ (for $i = 0, 1, 2, 3, \ldots$) imply that there are i requests in the system and the server is on at time t. In addition, let $X(t) = D$ denote that the server is down (and there are no customers) at time t. Clearly $\{X(t), t \geq 0\}$ is a CTMC with rate diagram shown in Figure 5. The CTMC is ergodic, and for $j = D, 0, 1, 2, \ldots$, let

$$p_j = \lim_{t \to \infty} P\{X(t) = j\}.$$

In order to obtain the steady state probabilities p_j, consider the following balance equations:

$$\begin{aligned}
\alpha(p_0 + p_1 + \ldots) &= \beta p_D \\
\beta p_D + \mu p_1 &= (\lambda + \alpha) p_0 \\
\mu p_2 + \lambda p_0 &= (\lambda + \alpha + \mu) p_1 \\
\mu p_3 + \lambda p_1 &= (\lambda + \alpha + \mu) p_2 \\
\mu p_4 + \lambda p_2 &= (\lambda + \alpha + \mu) p_3 \\
\vdots &= \vdots
\end{aligned}$$

From the first equation above, we have $p_D = \frac{\alpha}{\alpha+\beta}$. Multiplying the second equation by 1, third equation by z, fourth equation by z^2, and so on, and summing up:

$$\psi(z) = \frac{\mu p_0 - z\beta p_D - p_0 \mu z}{\mu + \lambda z^2 - \lambda z - \alpha z - \mu z}, \tag{2}$$

where $\psi(z) = p_0 + p_1 z + p_2 z^2 + p_3 z^3 + p_4 z^4 + \ldots$ (notice that $\psi(1) = 1 - p_D$). Standard techniques such as $\psi(0) = p_0$ and $\psi(1) = \beta/(\alpha + \beta)$ does not yield a solution for p_0. Hence we need the following Lemma to determine p_0 and hence $\psi(z)$.

Lemma 1 *Consider a series of real positive-valued numbers a_0, a_1, a_2, etc. For any function of the form $\phi(z) = \sum_{i=0}^{\infty} a_i z^i$ such that $\phi(z)$ can be written as a fraction $\phi(z) = \frac{A(z)}{B(z)}$, where $A(z)$ and $B(z)$ are polynomials, if $B(z^*) = 0$ for any $z^* \in [0, \infty)$ then $A(z^*) = 0$.*

Proof. By definition $\phi(z)$ is a continuous, differentiable and increasing function over $z \in [0, \infty)$. For some $z^* \in [0, \infty)$, let $B(z^*) = 0$. If $A(z^*) > 0$ (similar result can be shown for $A(z^*) < 0$), then $\phi(z^*-) = -\phi(z^*+)$ and $|\phi(z^*-)| \to \infty$. This contradicts the fact that $\phi(z)$ is continuous,

differentiable and increasing function over $z \in [0, \infty)$. Hence $A(z^*) = 0$. ■

We now use the above Lemma to derive a closed-form algebraic expression for $\psi(z)$ from Equation (2). The result is described in the theorem below.

Theorem 3 *The function $\psi(z)$ is given by*

$$\psi(z) = \frac{\mu p_0(1-z) - \frac{z\alpha\beta}{\alpha+\beta}}{\lambda z^2 - (\lambda+\mu+\alpha)z + \mu}, \tag{3}$$

where p_0 is given by

$$p_0 = \frac{\alpha\beta}{\mu(\alpha+\beta)} \left[\frac{\lambda+\mu+\alpha - \sqrt{(\lambda+\mu+\alpha)^2 - 4\lambda\mu}}{\lambda-\mu-\alpha+\sqrt{(\lambda+\mu+\alpha)^2-4\lambda\mu}} \right]. \tag{4}$$

Proof. Using the results from Lemma 1, setting the denominator of $\psi(z)$ in Equation (2) to zero, we get

$$z^* = \frac{(\lambda+\mu+\alpha) - \sqrt{(\lambda+\mu+\alpha)^2 - 4\lambda\mu}}{2\lambda},$$

as the unique solution such that $z^* \in [0, \infty)$. Setting the numerator of $\psi(z)$ in Equation (2) to zero at $z = z^*$, we get

$$p_0 = \frac{\alpha\beta z^*}{(\alpha+\beta)\mu(1-z^*)}.$$

By substituting for z^*, we get p_0 in Equation (4). Also, by rearranging terms in Equation (2), we get Equation (3). ■

Based on the above theorem, we can derive an asymptotic result. We let the server up and down times to be extremely large (especially in comparison to the arrival and service rates). Then we get the following remark:

Remark 1 *If $\lambda < \mu$, $\alpha \to 0$ and $\beta \to 0$ such that $\frac{\alpha}{\beta} \to r$, then $p_D = r/(1+r)$ and for $i = 0, 1, 2, \ldots$, $p_i = (1-p_D)(1-\lambda/\mu)(\lambda/\mu)^i$.*

The above result confirms our intuition that the system reaches steady state when the server is up. Therefore, the probability that there are i requests is the product of the probability that the server is up, and the probability there are i requests in steady state, given the server is up.

We now derive the two key performance measures, namely, loss and response time. Let P_ℓ be the probability that a request is lost and let Δ be the average delay (or response time) at the server as experienced by users that receive a response. Note that the latter is a conditional expected value,

conditioned on receiving a response. The two QoS measures are P_ℓ and Δ, lower their values, better the QoS. Both measures are in terms of L, the time-averaged number of requests in the system (note that it includes the down times when there are no requests in the system). The following theorem summarizes the results:

Theorem 4 *The average number of requests in the web-server-system is*

$$L = \frac{1}{\alpha}\left[\frac{\lambda\beta - \mu\beta + p_0\mu(\alpha+\beta)}{\alpha+\beta}\right], \tag{5}$$

where p_0 is described in Equation (4). The QoS measures P_ℓ and Δ, in terms of L are given by

$$P_\ell = \frac{\alpha L(\alpha+\beta) + \lambda\alpha}{\lambda(\alpha+\beta)}, \tag{6}$$

$$\Delta = \frac{L(\alpha+\beta)^2}{\lambda\beta^2}. \tag{7}$$

Proof. By definition,

$$L = 0p_D + 0p_0 + 1p_1 + 2p_2 + 3p_3 + \ldots,$$

and clearly that can be written as $L = \psi'(1)$. By taking the derivative of $\psi(z)$ in Equation (3), and then letting $z = 1$, we get Equation (5). The number of requests that were dropped per unit time is $\alpha(1p_1 + 2p_2 + 3p_3 + \ldots) = \alpha L$. Therefore the fraction of requests that entered the queue and were dropped when the server switched from on to off, is $\frac{\alpha L}{\lambda(1-p_D)}$. The probability that an arriving request will complete service, given that it arrived when the server was up, is given by (conditioning on the number of requests seen upon arrival)

$$\sum_{j=0}^{\infty}\left(\frac{p_j}{1-p_D}\right)\left(\frac{\mu}{\mu+\alpha}\right)^{j+1} = \frac{\mu}{\mu+\alpha}\frac{1}{1-p_D}\psi\left(\frac{\mu}{\mu+\alpha}\right) = \frac{\mu}{1-p_D}\frac{\beta - p_0(\alpha+\beta)}{\lambda(\alpha+\beta)}.$$

Therefore the rate at which requests exit the queue is $\mu\frac{\beta}{\alpha+\beta} - \mu p_0$, which makes sense as whenever there are one or more requests in the system, the exit rate is μ. In addition, since the drop rate (derived above) is αL, we can write $\mu\frac{\beta}{\alpha+\beta} - \mu p_0 = \lambda(1-p_D) - \alpha L$, which again makes sense and the total arrival rate when web server is on is $\lambda(1-p_D)$.

We also have a fraction p_D requests that are rejected when the server is down. Therefore the loss probability is $\frac{\lambda p_D + \alpha L}{\lambda}$, and by substituting for p_D we obtain P_ℓ in Equation (6). Using Little's law, we can derive Δ in the following manner. The expected number of requests in the system when the server is on is $\frac{L}{1-p_D}$. Of these L, a fraction $\frac{\lambda(1-p_D)-\alpha L}{\lambda(1-p_D)}$ only will receive service. Therefore the average delay (or response time) at the server as experienced by users that receives a response is given by $\frac{L}{\lambda(1-p_d)^2}$ which is the expression in Equation (7). ■

5 Pricing Based on QoS

We derived algebraic expressions for the two performance measures: average delay for a request that receives a response (Δ) in Equation (7), and probability that a request is lost (P_ℓ) in Equation (6). Both expressions were in terms of α, β, λ and μ. The terms α and β are beyond the control of the three players in the system namely, users, client and web server. What changes QoS and price are λ and μ. In the next few subsections we illustrate the relationships between QoS and price in terms of λ and μ. Notice that $\mu = 1/S$, where S is the average service time defined in Equation (1). As discussed in Section 3.4, the caching scheme is selected only once, and thereby all terms except the processor speed (c) in Equation (1) are fixed. Throughout this section, varying μ means varying c.

5.1 User and Web Server Interaction

For a fixed server processing speed c (and hence fixed μ), by plotting Δ and P_ℓ against λ, one can see that both are increasing functions of λ. This is because as the load increases, performance deteriorates. On the other hand if the delay or loss increases, the user demand (in terms of λ) decreases and vice versa. The server performance and customer behavior is depicted in Figure 6.

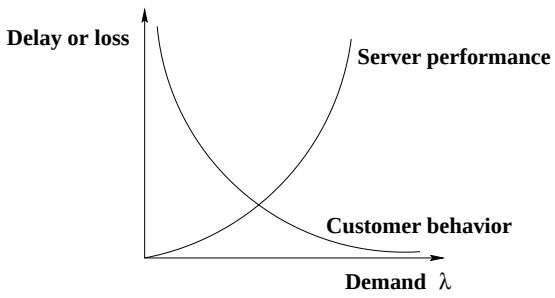

Figure 6: Server performance and customer behavior versus λ

The following two equations describe a functional relationship between λ and the performance measures from a customer behavior standpoint:

$$\lambda = cb_\ell(P_\ell), \tag{8}$$

$$\lambda = cb_d(\Delta). \tag{9}$$

Since there are two equations, for a given Δ and P_ℓ, the customer demand rate λ will be the minimum of the two possible values. In other words, $\lambda = \min\{cb_\ell(P_\ell), cb_d(\Delta)\}$. Although in

Section 5.3 we will assume a functional form for $cb_\ell(\lambda)$ and $cb_d(\lambda)$, in reality this need not be known, especially to the web server. In the iterative algorithm presented in Section 5.4, we will see how the functions need not be known to the server.

5.2 Client and Web Server Interaction

The two parties that are involved in economic transactions are the client and web server. The client would typically be willing to pay a higher price if it can attract a larger number of users to the web site. This is because the expected profit increases if number of users increases. This relationship is depicted in Figure 7. A functional relationship between the price a client is willing to pay (p_r) and λ can be thought as

$$p_r = g(\lambda). \tag{10}$$

The functional relationship in Equation (10) above need not be known to the web server as we will see in the iterative algorithm presented in Section 5.4. However, in Section 5.3 we will assume a functional form for $g(\lambda)$.

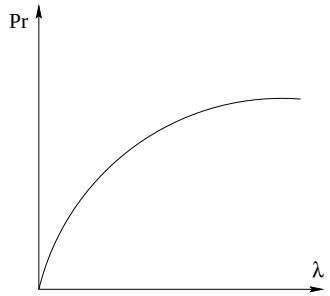

Figure 7: Price client is willing to pay versus λ

The web server on the other hand has several costs to recover including investment costs, maintenance costs, operational costs, personnel costs, etc. However these costs are fixed with respect to μ or c for that matter. One cost that has been ignored in the past that researchers are paying attention to, is power cost. Typically c, the processing speed can be modified by suitably changing the voltage. Since power is proportional to the square of the voltage, the cost incurred can be assumed to be quadratic. In particular, the relationship between the price the client is willing to pay (p_r), and the processing speed (c), is given by

$$p_r = k + ac^2, \tag{11}$$

where k is the fixed cost and a is the cost associated with power. This relationship is depicted in

Figure 8. The functional relationship in Equation (11) must be known to the web server in order to decide on an acceptable price.

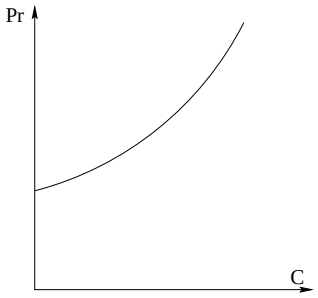

Figure 8: Price client is willing to pay versus c for server

5.3 Numerical Example

In this section we present a numerical example for the various parameters of the problem as well as functional relationships. There are two phases of execution, the initial phase and the iterative phase. In the initial phase, caching decisions are made, cache miss probability is computed as well as user arrival rates are obtained. For this the server uses a nominal (possibly mid-range c value) for processing speed. Using Equation (1), the service rate μ is obtained. For our numerical values, we obtained a relationship between c and μ as follows:

$$\frac{1}{\mu} = \frac{15000}{c} + 0.035,$$

from which an initial value of μ can be obtained. In the iterative phase, using the arrival rate λ, for various c values, the server computes the loss probability (P_ℓ) and delay (Δ) using Equations (6) and (7). For the calculations, $\alpha = 0.02$ and $\beta = 0.1$ were used. The server also computes the price for various c values using numerical values for Equation (11):

$$p_r = 10 + 30(c/1000000)^2.$$

The server presents (based on several c values), various options for price (p_r) and QoS (P_ℓ and Δ). Based on the λ value given to the client by the server, the client uses Equation (10) to determine the price (p_r) the client is willing to pay. The numerical values chosen for Equation (10) is:

$$p_r = 12.65\sqrt{\lambda}.$$

Using this p_r, the server now computes the corresponding c (from Equation (11), i.e. $p_r = 10 + 30(c/1000000)^2$) and uses it for the next iterative stage. At the end of the iterative stage the new

λ values are obtained according to the following customer behavior functions, corresponding to the numerical values in Equations (8) and (9):

$$\lambda = \min(1.195/\Delta, 1.8/P_\ell).$$

This iterative phase is performed continuously.

For this numerical study, it turns out the iterative phase does not converge. Instead it oscillates between two price values, irrespective of whether we start with an initial phase with a smaller price (Figure 9), median price between the two values (Figure 10) or larger price (Figure 11). However for other numerical examples, the results range from cases where the price converges to a single fixed point, to cases where the oscillating price values depend on the initial value of λ and c.

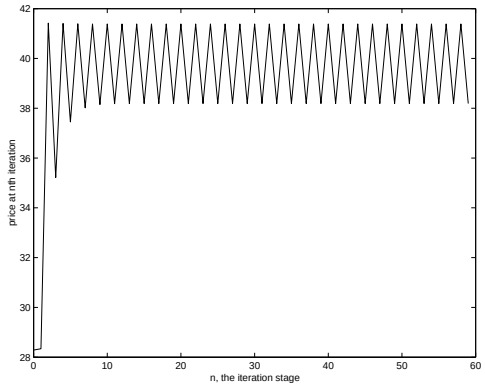

Figure 9: Price across iterations, initial values include: $\lambda = 5$ and $c = 230770$

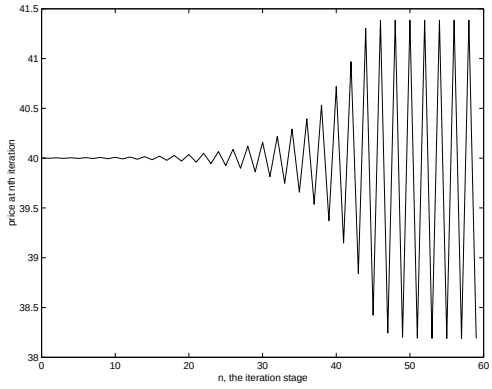

Figure 10: Price across iterations, initial values include: $\lambda = 10$ and $c = 1000000$

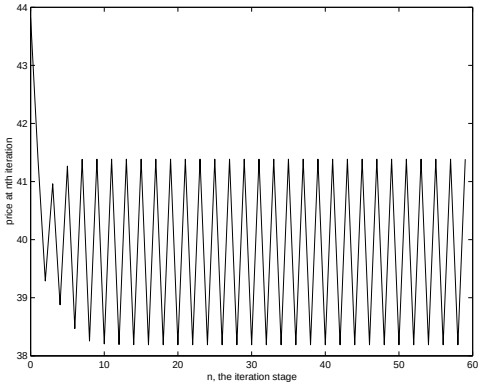

Figure 11: Price across iterations, initial values include: $\lambda = 12$ and $c = 2250000$

5.4 Implementation Details: Iterative Procedure

In the previous subsection we alluded to the possibility of observing chaotic behavior when a fixed point iteration is used (see Figures 9, 10 and 11). However it is important to realize a few things: the functional relationships are usually unknown, a single simple function does not typically characterize some of the relationships, other external factors steer the iterations in different directions, the processing speed c usually takes 8-10 discrete values (as opposed to our continuous assumption), and other incentives could be thrown in to steer away from the chaotic patterns.

In that light, we discuss some of the implementation details. Firstly from a user standpoint, it is assumed that the users are extremely sensitive to the changes in QoS. In practice, the web servers should wait for a reasonably long time before offering a new price to the client so that the users can first modify their arrival rate and stabilize to the new value. Secondly, the clients who are out of the picture in terms of request and response, would be interested in knowing if the customers (i.e. users) were satisfied. One way is to conduct surveys. However an easier method that the clients adopt is to send test messages (as dummy requests) periodically to evaluate the server performance. Finally, the web server uses algorithm PRICE described below for iteratively obtaining an optimal price.

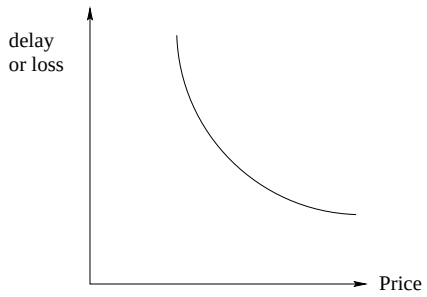

Figure 12: Price versus performance advertised by server

Algorithm PRICE

Input: α, β, k, a
Output: Equilibrium price
Method:

1. Initially server must charge a base price for a period.
2. Based on user arrival patterns, λ is obtained by server. In addition, server determines optimal caching policy using the estimated probabilities of accessing various documents.
3. Web server presents the value of λ to client, and hence offers a graph (or table) of price versus performance as shown in Figure 12.
4. The client determines the most suitable price it can pay for a given λ, and offers that price. Hence the web server knows the performance to provide and thereby c.
5. Server adjusts c suitably and runs the system for a period of time. Based on the QoS obtained, the user demand rate λ changes suitably.
6. Server records new arrival rate λ.
7. Steps 3, 4, 5 and 6 are solved iteratively until λ, c (hence μ) and price are stabilized.

6 Concluding Remarks

In this research we concentrate on a customized pricing policy that a web server offers to each of its clients. This pricing policy is a function of the QoS (or performance) experienced by the clients' clients, i.e. the users. We first study two different caching policies namely LRU and LFU, and evaluate their performance and applicability. We assume that the caching decision is a strategic one-time decision. Next, we obtain closed-form algebraic expressions for performance measures such as response time and loss experienced by users. In this, one of the innovative components is to model congestion in the web server farm by assuming the web server to be down. Then we study a pricing model where we assume users are sensitive to the QoS they receive (such as

response time and request loss). To adjust to the fluctuating demand, the server can suitably modify the processing speed from time to time. Based on the dynamics between the three players (users, clients and web servers), we show that the price and web server settings could either stabilize or oscillate between several points (possibly depending on the initial condition) indicating chaos.

This paper is certainly a preliminary one and several extensions are possible in the future. First of all the user arrival rates λ change significantly over time. In addition, the congestion episodes are correlated with periods of high arrival rates. It is also important to incorporate some admission control (possibly based on pricing) on the user traffic. Although the exponential distribution produced nice closed-form results, it may be prudent to consider other distributions. In terms of the web caching mechanisms, other schemes that are more dynamic can be used. In terms of dynamically executing control, one other parameter of interest is to increase the number of servers during periods of heavy traffic. For example, if there are sites that have unusually high demand for a short time in a day, only during those times, the queue of requests can be served by multiple servers. All these have a tremendous impact on pricing, which could also be time-of-the-day dependent.

Acknowledgements

The author thanks the anonymous referees for their comments and suggestions, which led to considerable improvements in the content and presentation of this article. This research was partially supported by the following NSF grants under the ITR initiative: ACI-0325056 and ANI-0219747.

References

Adler, M., J.-Y. Cai, J. Shapiro, and D. Towsley (2003). Estimation of congestion price using probabilistic packet marking. In *IEEE INFOCOM*, pp. 2068–2078.

Breslau, L., P. Cao, L. Fan, G. Phillips, and S. Shenker (1999). Web caching, and zipf-like distributions: Evidence, and implications. In *IEEE INFOCOM*.

Caesar, M., S. Balaraman, and D. Ghosal (2000). A comparative study of pricing strategies for ip telephony. In *IEEE GLOBECOM*, pp. 344–349.

Chao, X. (1995). A queueing network model with catastrophes and product form solution. *Operations Research Letters 18*(2), 75–79.

Chao, X. and Y. Zheng (2003). Transient analysis of immigration birth-death processes with total catastrophes. *Probability in the Engineering and Informational Sciences 17*, 83–106.

Cocchi, R., D. Estrin, S. Shenker, and L. Zhang (1993). Pricing in computer networks: Motivation, formulation and example. *ACM/IEEE Transactions on Networking 1*, 614–627.

Courcoubetis, C. and R. Weber (2003). *Pricing Communication Networks: Economics, Technology and Modelling*. Chichester: Wiley.

Crowcroft, J. (1996, 11). Pricing the internet. *IEE Colloquium on Charging for ATM* (222), 1–4.

Davies, G., M. Hardt, and F. Kelly (2004). Come the revolution - network dimensioning, service costing and pricing in a packet switched environment. *Telecommunications Policy 28*, 391–412.

Dotster (2004). *http://www.dotster.com/webhosting/webhosting_commerce.php*.

Edell, R. J., N. McKeown, and P. P. Varaiya (1995). Billing users and pricing for tcp. *IEEE Journal on Selected Areas in Communications 13*(7), 1–14.

Fortune-City (2004). *http://www.fortunecity.com/business-web-hosting-plans.shtml*.

Global-Servers (2004). *http://www.globalservers.com/*.

Gray, J. and D. Siewiorek (1991). High-availability computer systems. *Computer 24*(9), 39–48.

Hamadeh, I., Y. Jin, S. Walia, G. Kesidis, and C. Kirjner (2004). Pricing and security issues for residential broadband access. In *CISS*.

Henderson, T., J. Crowcroft, and S. Bhatti (2001). Congestion pricing: paying your way in communication networks. *IEEE Internet Computing 5*(5), 85–89.

Jelenkovic, P. R. and A. Radovanovic (2004). Optimizing lru caching for variable document sizes. *to appear in Combinatorics, Probability and Computing*.

Jin, N. and S. Jordan (2004). The effect of bandwidth and buffer pricing on resource allocation and qos. *Computer Networks, Special Issue on Internet Economics: Pricing and Policies 46*(1), 53–71.

Kelly, F. P. (1996). *Charging and Accounting for Bursty Connections*. Cambridge, MA: MIT press.

Kumar, B. and D. Arivudainambi (2000). Transient solution of an m/m/1 queue with catastrophes. *Computers & Mathematics with Applications 40*, 12331240.

Liu, Z., N. Niclausse, and C. Jalpa-Villanueva (2001). Traffic model and performance evaluation of web servers. *Performance Evaluation 46*, 77–100.

MacKie-Mason, J. K. and H. Varian (1995). *Pricing the Internet*. Cambridge, MA: MIT Press.

Menasce, D. and V. Almeida (1998). *Capacity Planning for Web Performance*. PTR: Prentice Hall.

Mookerjee, V. and Y. Tan (2002). Analysis of a least recently used cache management policy for web browsers. *Operations Research 50*(2).

Odlyzko, A. (2000). Should flat-rate internet pricing continue. *IT Professional 2*(5), 48–51.

Parris, C., S. Keshav, and D. Ferrari (1992). A framework for the study of pricing in integrated networks. In *Technical Report TR-92-016, International Computer Science Institute*.

Shenker, S., D. Clark, D. Estrin, and S. Herzog (1996). *Pricing in Computer Networks: Reshaping the Research Agenda*.

Turner, D. and K. Ross (2004). Lightweight currency paradigm for the p2p resource market. In *Seventh International Conference on Electronic Commerce Research*.

Website-Providers (2004). *http://webhosting.websiteproviders.net/guarantees/quality_of_service/*.