Pricing Interactive Computer Services

William A. Gale and Roger Koenker
Bell Laboratories, Murray Hill, New Jersey 07974, USA

This paper illustrates a method for setting theoretically and empirically defensible prices for interactive computer services. The theoretical rationale is that prices should reflect the marginal social cost for using interactive computing measured by the delay imposed on other users. The empirical basis for the prices is the direct measurement of delay as a function of load, and of average load levels as a function of time. An intrinsic problem of shared resources such as interactive computer systems is congestion. Every user contributes to some degree to the general deterioration of service quality (response time) as system loads increase. Since capital and operating costs of computer systems are fixed irrespective of system load, the delay costs imposed on users by congestion are the predominant *marginal* costs of system operation. It is an economic theorem that unless prices are set to marginal social costs, inefficient use of resources results. Therefore, efficient use of interactive computing services requires setting prices to equal the expected cost of marginal congestion delay. The paper shows that under plausible conditions, marginal congestion prices will cover costs of operation. Furthermore, profitability with such prices in effect gives information on when to make additional investment in capacity. We also show how to construct statistical models of response time from direct observations. Direct observations can provide a robust and understandable empirical basis for marginal congestion pricing. The empirical work needed is illustated by measurements on a VAX 11/780 offering UNIX™ service. We show the use of several statistical techniques to display the results understandably.

1. INTRODUCTION

This paper is about a method for pricing interactive computer services. We believe that methods now in use are socially inefficient and that better pricing methods would lead to both greater productivity and to greater fairness to users.

Our basic observation is that the capital and operating costs of an interactive computer system are approximately independent of short run variations in the load on the system. This is quite clear for the capital costs. Purchase of a machine provides the same capacity at 3 a.m. as at 3 p.m. whether or not that capacity can be used. For a modern interactive computer system which uses disk files rather than tape files, and which is capable of running unattended for substantial periods of time, the operating costs have also become largely independent of load. It might still be argued that maintenance costs are necessarily proportional to load. We believe the remaining operating costs that are dependent on load are small compared to the costs considered below, and have built models on this assumption.

The more common approach appears to be to observe that a permanent increase in load requires additional investment in capacity and to allocate the capital costs to specific components of load. A substantial portion of the literature on pricing computer services takes this approach (see the survey by McKell, Hansen and Heitger[1]). This approach can only accidentally result in socially efficient prices. It typically results in prices that are both inefficient and unjust. Nor is it necessary to ensure recovery of capital costs.

The most important social cost which does vary with the load is the cost of congestion in sharing the resource. In an interactive computer system, the resources of the machine are devoted to different tasks in rapid succession so that no task need wait long for some service. This means that if one user creates an additional load, it will necessarily create extra delay for all other users. Given the salaries of people who use computer services, these congestion costs are not small. During peak hours on the system we have studied, we estimate the total value of waiting time at 286$/h. Of this, 122$/h is the total user waiting for their own jobs, and 164$/h is the cost of waiting on others jobs. These costs can be compared to the 56$/h revenue collected during these times.

Naturally, users are familiar with the impact of congestion on themselves, and to some extent react to the expected congestion levels. However, barring an assumption of altruism, the user is thus only accounting for a small fraction of the total congestion cost—the part affecting him or her personally. Altruism may be effective if only a few people use a given system and if they are working on a common task. Ordinarily these conditions do not hold for interactive computer services, and it is for this reason that simply providing better information on what congestion is expected does not lead to efficient use.

Rather, it is a well established economic principle that if individuals must pay an amount commensurate with the costs they impose on others, individually rational behaviour will lead to socially efficient results. The basic fairness of this principle is also evident. Lerner[2] provides a very lucid explanation of this view. The costs one user imposes on others are the marginal costs of congestion. Therefore we term the proposed pricing method marginal congestion pricing.

2. THEORETICAL RATIONALE

Marginal congestion pricing rests on a sound economic basis. The purpose of this section is to illustrate the principles of reasoning by developing a simple model. The model supposes a simple computing environment with scalar measures of system load, q, and system

CCC-0010-4620/84/0027-0008 $05.00

capacity k. If cpu time is the most important single load dimension, the units of these scalar measures might be in cpu seconds per second. The private cost as seen by an individual user generating an increment dq of system load is composed of two parts: a pecuniary part and a temporal part which we express as,

$$p(dq) = r\,dq + vw(q, k)\,dq \qquad (1)$$

where r denotes the money price of the service q, $w(q, k)$ is the wait incurred at system load q and system capacity k for a unit of system load, and v is the value of waiting time to users.* We will assume throughout that behaviour is influenced only by *expected* waiting time and that this expectation is a smooth, known function of system load and capacity levels. The *social* cost of doing dq is simply the value of its initiator's wait plus the value of other users' incremental congestion delay,† which may be expressed as,

$$s(dq) = vw(q, k)\,dq + qv\frac{\partial w}{\partial q}\,dq \qquad (2)$$

If $p(dq) < s(dq)$ then individual users are apt to initiate tasks like dq which may be justifiable in terms of their private costs, but whose benefits do not exceed their social costs. As a consequence too much congestion will occur. On the other hand, if private costs exceed social costs, some tasks whose benefits exceed their social costs will not be initiated and the service will be underused.

The discrepancy between private and social costs is obviously resolved if the following pricing policy is adopted: set

$$r = vq\frac{\partial w(q, k)}{\partial q} \qquad (3)$$

Under this policy, private incentives coincide with social incentives, and efficient use of the shared resource occurs.‡ Our pricing rule is a thinly disguised variant of the [price = marginal cost] rule which pervades economic reasoning. However, marginal costs are borne in this case by other consumers rather than directly by the producer of the service as is usually the case. Pigou[4] developed the essential ideas of marginal congestion pricing in the context of highway tolls, see also Ref. 5. Naor[6] and Marchant[7] have studied the theory of congestion pricing in queueing models with and without priority classes.

Tables 1 and 2 may give some feeling for the magnitudes of the numbers involved in Eqn (3). The tables show the results of considering each of three resources cpu, i/o, and memory in turn as the single resource described by the model of this section. The entries give the total value for all users of revenue collected based on the resource, and of the value of user waiting time. The waiting time is split into two components according to Eqn (2). The columns with totals give an approximate overall picture

Table 1. Congestion costs and revenues—afternoon

		CPU $/h	I/O $/h	Memory $/h	Total $/h
Revenue	rq	56	0	0	56
Own wait	vwq	67	43	12	122
Others wait	$vq^2\,\partial w/\partial q$	98	57	9	164
Private cost	Revenue + own wait	123	43	12	178
Social cost	Own + other waits	165	100	21	286

Table 2. Congestion costs and revenues—early morning

		CPU $/h	I/O $/h	Memory $/h	Total $/h
Revenue	rq	1.40	0	0	1.40
Own wait	vwq	2.00	2.88	3.40	8.28
Others wait	$vq^2\,\partial w/\partial q$	0.12	0.24	0.34	0.70
Private cost	Revenue + own wait	3.40	2.88	3.40	9.68
Social cost	Own + other waits	2.12	3.12	3.74	9.98

of revenue and costs. (The amounts neglected by this simple approach can be seen graphically in Fig. 9.)

It can be seen in Table 1 that waiting time costs are substantial and that private cost was less than social cost by a factor of about two. It can also be seen that cpu is the single most important resource, and i/o is also important, but memory could be neglected. These results apply to a particular machine, and may not be general. It is probably reasonable to study all resources infrequently to establish which resources to study closely more often. Table 2 shows a better balance of private and social costs, but mostly because the value of own waiting time is the bulk of the cost in the early morning. The monetary price being charged is a factor of ten above the cost imposed on other users.

A geometrical expression of the theory can also be given, as in Fig. 1, which shows the situation for cpu time in the afternoon during a prior period of rates on the machine we studied. The demand curve labelled $D(p)$ may be regarded as the (hypothetical) demand which would arrive if there were no perceptible processing delay. The demand curve $d(r)$ describes the dependence of demand upon the pecuniary price, r, accounting for the effect of the congestion delay. At any r, $d(r) \leq D(r)$ since some demand is suppressed by the additional cost imposed by the congestion delay. As demand approaches capacity and the congestion becomes large the discrepancy between $d(r)$ and $D(r)$ becomes larger.

At the current price* of $46.50/cpu hour the peak demand is roughly 0.56 cpu h/h. Thus revenue is given by the lower rectangle in the Figure, whereas the upper rectangle labelled 'congestion' depicts the *value* of time lost due to congestion delay. Had there been enough capacity to yield a negligible congestion delay, $r \approx \$160$ could have been charged and the congestion effect could have been entirely appropriated as revenue. The triangle labelled 'surplus' is a measure of the excess in users' benefits, or willingness to pay, above and beyond the costs, both pecuniary and temporal, which they incur.

* The assumptions of (i) scalar load and capacity and (ii) homogeneity of v across users may be easily relaxed.[3]

† We assume at this point that an increment in load causes no increase in the direct operating costs of the system. Such costs could be easily incorporated, but for most interactive computing environments the capital and operating costs of the machines are fixed, essentially independent of the load.

‡ Here we assume that individual users are price-takers, and either cannot or do not respond as a coalition to the announced pecuniary price r.

* Section 5 discusses our method of linearizing the study period pricing policy which was non-linear in memory usage.

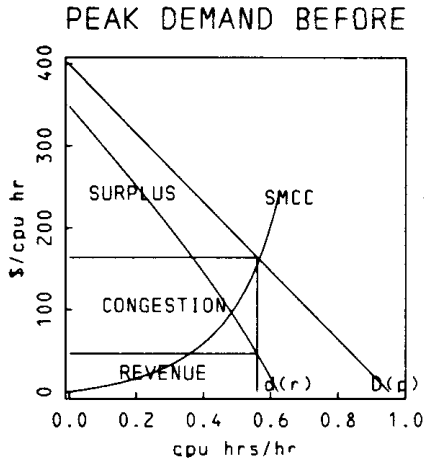

Figure 1. Marginal congestion pricing graphically.

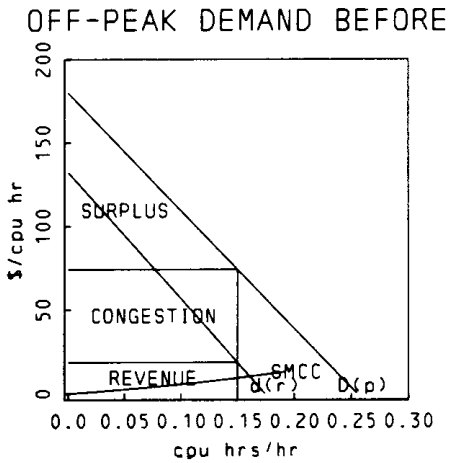

Figure 3. Marginal congestion pricing graphically.

The upward sloping curve in Fig. 1 labelled SMCC is our estimate of the social marginal congestion cost, $vq\ \partial w(q)/\partial q$, as a function of cpu load. Here, and elsewhere we have set v, the value of time, to \$50/h. At the current peak load of 0.56 cpu h/h another cpu hour of load creates roughly \$160 of additional congestion delay to other users, yet the perceived price is less than one third this amount. Some activities which appear worth while at the latter price may well appear frivolous at the former, which reflects the real costs imposed on the user community. Equating the pecuniary price and the social marginal congestion cost gives Fig. 2, in which demand is suppressed slightly, revenue is considerably larger, and congestion is considerably reduced. Now users are forced to pay the full value of the delay costs they impose on others.

In contrast, we illustrate the current off-peak (late evening) situation in Fig. 3. The cpu load is now only 15% of capacity despite a 60% evening price discount. Social marginal congestion costs are negligible at this low load, and therefore we have the opposite difficulty to that illustrated previously. The perceived price to users is much higher than the real costs imposed by computing tasks and thus there may be worthwhile activities which are discouraged by misleading price signals. Figure 4 illustrates a rectified off-peak situation in which the price is equated to social marginal cost, revenue falls, congestion increases slightly, and the evening utilization increases.

2.1 Capacity planning and cost recovery

To this point we have said nothing about the direct costs of owning and operating the physical resources which provide the computing services under study. It may seem unseemly, even fiscally irresponsible, to promulgate pricing policies without explicitly considering these costs. How do we know that a system operated with our proposed pricing rule can break even? To answer this question we must extend our simple model slightly to explicitly incorporate capacity choice.

Suppose the waiting time function is homogeneous of degree zero in load and capacity, i.e.

$$w(z, k) = w(\lambda q, \lambda k), \quad \lambda > 0 \tag{4}$$

This seems plausible if capacity is expanded by adding essentially independent machines. Further, assume that capacity costs are linear,

$$c(k) = \rho k \tag{5}$$

If the preceding two quite plausible conditions are met, then not only can the operation of the system break even, but the break-even condition becomes a crucial investment signal.

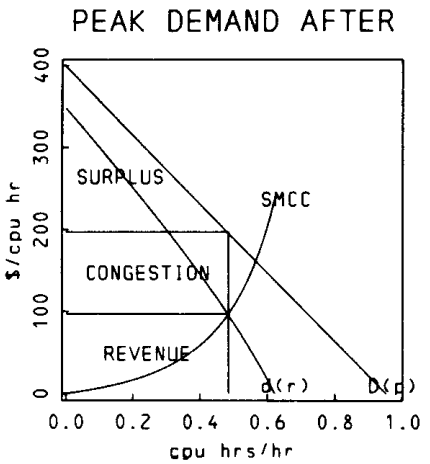

Figure 2. Marginal congestion pricing graphically.

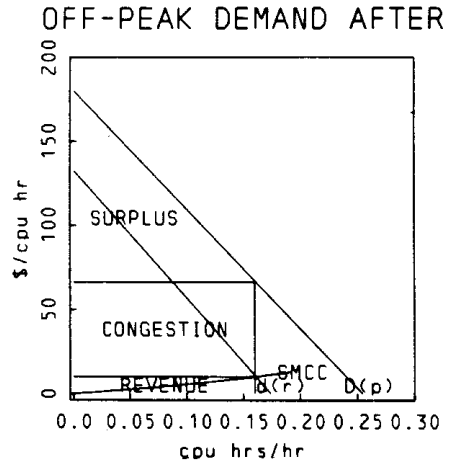

Figure 4. Marginal congestion pricing graphically.

A socially efficient investment policy would adjust capacity to equate the value of an incremental reduction in user waiting to the price of an additional unit of capacity, i.e. it would choose k so that,

$$-qv\frac{\partial w(q, k)}{\partial k} = \rho \tag{6}$$

(Again we make differentiability assumptions for convenience.) In terms of Figs 1–4 expanding capacity has the effect of shifting $d(r)$ towards $D(p)$. We would like to do this until the savings in the congestion rectangle are exceeded by the cost of the new capacity. The rate of profitability of the system is simply

$$\pi = rq - \rho k \tag{7}$$

which under our proposed pricing rule becomes

$$\pi = q^2 v\frac{\partial w(q, k)}{\partial q} - \rho k \tag{8}$$

but (4) implies

$$q\frac{\partial w(q, k)}{\partial q} + k\frac{\partial w(q, k)}{\partial k} = 0 \tag{9}$$

so we have

$$\pi = -qkv\frac{\partial w(q, k)}{\partial k} - \rho k \tag{10}$$

Hence $\pi = 0$ is equivalent to the equilibrium capacity condition (6). If $\pi > 0$, then

$$-qv\frac{\partial w(q, k)}{\partial k} > \rho \tag{11}$$

which implies that an increment in k costs less than the value of its resulting reduction in user waiting time. Thus, positive profit is a signal to expand capacity. Conversely, negative profit is a signal to contract capacity. Not only does the marginal cost pricing rule (3) induce individual users of the service to behave efficiently, but via the profitability of the system operation it provides an informative signal of socially productive investment opportunities. When conditions (4) and (5) are violated then the profit signal becomes somewhat ambiguous. For example if (4) holds but $c(k)$ is concave (suppose, for example, that it is linear in k but has a positive fixed cost) then $\pi > 0$ is sufficient but not necessary for capacity expansion.

3. EMPIRICAL BASIS: CONGESTION

Extending the simple scalar model to the much more realistic context of vector-valued load and capacity variables is quite straightforward. Let $\mathbf{q} \in \mathbf{R}^n_+$ denote an n-vector of load variables which might be distinguished by generic tasks, time of submission or any number of other considerations. Let $\mathbf{k} \in \mathbf{R}^m_+$ denote an m-vector of capacity variables. The function $w: \mathbf{R}^n_+ \times \mathbf{R}^m_+ \to \mathbf{R}^n_+$ maps load and capacity configurations into a vector of waiting times whose elements correspond to each of the load variables.

Again, equating private and social (marginal) costs leads to the pricing rule

$$r_i = v\sum_{j=1}^{n}\frac{\partial w_j(\mathbf{q}, \mathbf{k})}{\partial q_i}q_j, \quad i = 1, \ldots, n \tag{12}$$

This pricing rule, which is a vector analogue of (3) gives prices r_i for loads of different kinds such as cpu, i/o and memory. An individual with usage q_i of each load type would pay

$$\sum_{i=1}^{n} r_i q_i \tag{13}$$

This is an easily understood means of pricing and it is easily administered since it only requires keeping a sum for each user of the total use of each kind of resource.

Equation (12) gives a rule for establishing r_i as a function of

v—the value of user's time
q_i—the total system load of type i
$\partial w_j/\partial q_i$—the incremental waiting time for resource j as load i increases.

The value of user's time can be estimated from an average salary. To illustrate, we use \$50/h, which is a representative loaded salary in our environment. The loading reflects support services provided to technical employees by others. The total system load is commonly measured in computer installations. We return to an analysis of it in the next section. What is not immediately available is the factor $\partial w_j/\partial q_i$. In this section we show how it can be obtained.

3.1 Experimental techniques

Although we are interested in the delay caused by an incremental job, we can most easily measure the delay to a test job as a function of the concurrent background. We will then be able to estimate the delay to our test job caused by an increment in the background load.

It is desirable that the resources required by a test job be independent of the load on the machine. In a complex system, this may not be possible, but some jobs will be more dependent than others. For instance, a job that checks how many users are present will have more to do when there is a greater load, and hence is undesirable. Likewise, a program that can be used by re-entering may have less to do when the system is busy, because the code may not need to be loaded. Because unexpected load dependencies like these are most likely for standard system utilities, creating small test programs is a reasonable approach.

To facilitate extending the measured results to a typical load, we selected test jobs that exercised primarily one computer resource. After preliminary study of the execution time of the test jobs, we found that most (90%) of the variance in execution time could be explained by concurrent use of two resources: cpu and disk access. Realizing that a large price was placed on memory use we decided to study memory closely too. Accordingly we built test jobs that emphasized cpu use, disk use, and memory use. The cpu intensive job counts for one second on an unloaded machine. Similarly, the disk intensive job copies files for one second. The memory job gets several hundred kilobytes of memory then sleeps, waking every second for ten seconds.

We used these test jobs with two different kinds of backgrounds. The first background was the daily load.

Under system control the test jobs were run every fifteen minutes, measurements taken, and the results appended to a file. The test jobs were run a random number of minutes following each quarter hour, the random number uniformly distributed on [0,9]. The second background type was of synthetic loads, each composed of multiple copies of one of the test jobs. These loads were generated between 3 a.m. and 6 a.m. when the activity on the system is normally extremely low.

With either kind of background, the information gathered for each test job was the elapsed time it took to run, and the total use of a variety of system resources while the test job was running. These included cpu use by other users and by the system, normal disk access, disk access for swapping, characters transferred on data lines, and references to system tables.

These measurements are made by *timex*,[8] a UNIX system utility. Since memory use is not measured by timex, it was measured immediately before each test job ran, using a program that examined system memory tables. The measurements we used for the daily background load were made during a fifteen week period in 1981.

3.2 Functional form

The following arguments gave us fairly strong expectations about the form of dependence of elapsed time for a test job on concurrent load. Consider a very simple case first: a machine that has but a single resource. Suppose it takes an average of τ seconds to respond to each request for the resource. A test job requires Q_0 requests for the resource and, while it is running, a total of Q other requests are made. Then the elapsed time, w, for the test job to run will be

$$\begin{aligned} w &= \tau Q_0 + \tau Q \\ &= w_0 + \tau Q \end{aligned} \tag{14}$$

where w_0 is the time for the test job to run when no other load is present. We measure the concurrent requests, and equations of this form could be used for estimation. However, a model that specifies the rate of arrival of requests, rather than the total requests in some period is more useful analytically. If requests arrive at the average rate q per second, then Q depends on w as $Q = qw$ so

$$w = w_0/(1 - \tau q) \tag{15}$$

Thus the linear Eqn (14) for elapsed time as a function of concurrent requests implies a non-linear dependence of elapsed time on a request rate. The form of (15) requires that as $q \to 1/\tau$ the elapsed time increases without bound. Thus $1/\tau$ could reasonably be interpreted as the capacity of the system—the maximum requests per second that it could handle.

The literature on queueing gives a more rigorous and broader rationale for the same functional form. The expected waiting time in the classical $M/M/1$ queue is given by

$$w = \frac{1/\mu}{1 - \lambda/\mu} \tag{16}$$

where λ is the rate of arrivals, and $1/\mu$ is the expected service time. For this case, the expected service time for the test job was $w_0 = 1/\mu$, and we can also identify $\tau = 1/\mu$ and $q = \lambda$ to relate this result to the simple exposition above. Under a processor sharing discipline, a small test job taking $w_0 < 1/\mu$ service time would also satisfy (15) in an $M/M/1$ queue. A much broader class of queues that will give a similar result are the networks of queues described by Jackson,[9] see Ref. 3.

Thus for a job testing resource j we have considerable reason to expect a dependence such as

$$w_j = \frac{w_{j0}}{1 - \tau_j q_j} \tag{17}$$

The models cited predict no dependence of waiting time for a job test resource j on use of resources other than j. However, in a real system, it seems prudent to consider this hypothesis empirically. We wish to introduce into Eqn (17) terms to test whether the waiting time for the jth resource depends only on the rate of use of that resource. It is natural to introduce such terms additively, giving

$$w_j = w_{j0} \Big/ \left(1 - \sum_{i=1}^{M} \tau_{ij} q_i\right) \tag{18}$$

In Eqn (18), the parameters of greatest interest are the τ_{ij}, the marginal effects of load of type i on response of type j. These parameters can be conveniently estimated without bias in the linear form

$$1/w_j \equiv \beta_{0j} + \sum_{i=1}^{M} \beta_{ij} q_i + \textit{error} \tag{19}$$

where $\beta_{0j} = 1/w_j$ and $\beta_{ij} = \tau_{ij}/w_j$. We use the term *marginal thruput* for the reciprocal of the elapsed time for a test job. From estimates of (19) we may compute estimates of

$$\partial w_j/\partial q_i = \tau_{ij} \Big/ \left(1 - \sum_{k=1}^{n} \tau_{kj} q_k\right)^2 \tag{20}$$

where the q_k are now interpreted as expected system loads.

3.3 Presentation of results

This section illustrates the use of partial residual plots to present the results of a multiple linear regression. It also shows that the system performance may change over time, requiring caution in making estimates. Our data span a quarter of a year, and are too numerous to display in complete detail. We will show detailed plots for the week of 6 April 1981, and summaries for other weeks. We have examined similarly constructed plots for several other weeks and months, and have found their appearance to be similar to those shown.

We have defined marginal thruput as the reciprocal of the waiting time for a test job. We have considered marginal thruput as a function of three variables representing loads on three different facilities. The load on the central processing unit (cpu) is measured in cpu seconds per second. In practice it is noted by the operating system every 1/60 second whether the cpu is busy or waiting. The cpu load rate is the fraction of 'busy' observations of all observations. The load on disk access equipment is measured in input or output counts per second. One count represents the transfer of one 512 byte block of data from or to the disk. The memory load is conceptually the number of byte seconds/second. In

practice we measure the bytes in use at one point of time shortly before a test job runs. Although we included many other variables in original exploratory work, these variables explained the bulk of the variance in our observations of response time.

Figures 5–7 show the dependence of marginal thruput for cpu test probes on each independent variable through partial residual plots. A partial residual plot gives a view of dependence of the dependent variable on one among several independent variables by controlling for the influence of all other independent variables.[10] The method of controlling is to regress the dependent variable as well as the chosen independent variable on the set of other independent variables, taking the residuals in each regression to represent the part of the variable not explained by the other variables. The partial residual plot then is one set of residuals plotted against the other. The coefficient from a simple regression fitted to such a plot is algebraically the same as that for the chosen independent variable in the multiple regression including it and all the variables controlled. The plot can then be

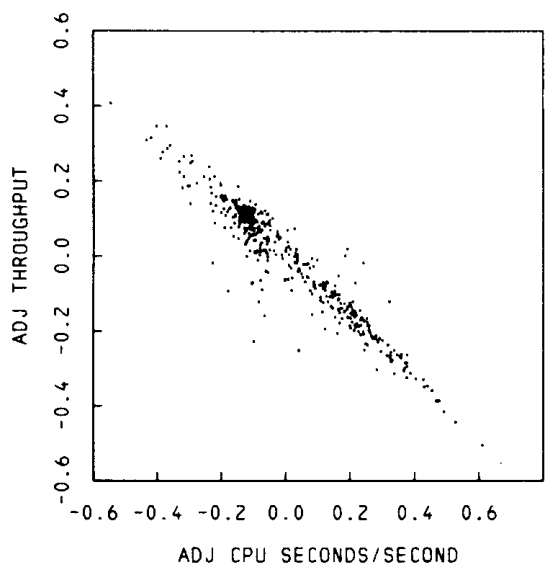

Figure 5. Partial residual plot: cpu wait vs. cpu load.

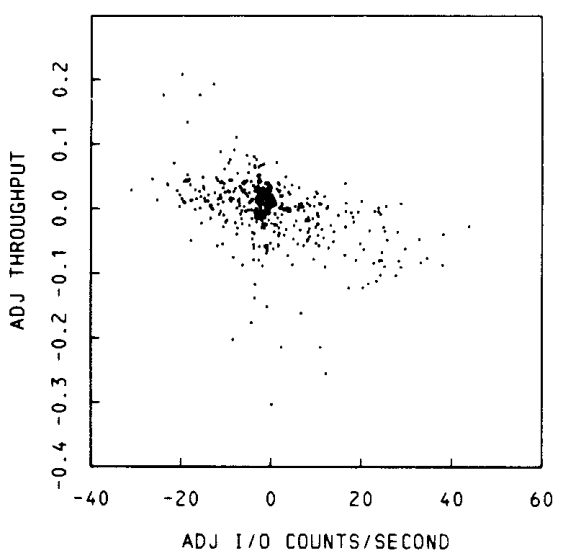

Figure 6. Partial residual plot: cpu wait vs. i/o load.

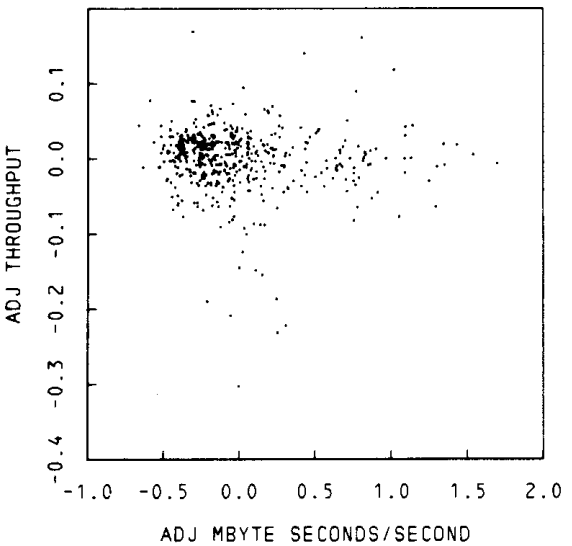

Figure 7. Partial residual plot: cpu wait vs. memory load.

interpreted as an ordinary scanner plot to check visually for linearity, outliers, and influential subsets of data.

In Fig. 5, showing the dependence of the cpu intensive probes on cpu load, a linear dependence is clear. The slope of the line is negative, giving a positive τ_m, as expected. Figure 6 shows some relationship. The impression of greater dispersion is primarily an artifact of the smaller range of the adjusted marginal thruput in this plot, but there is still a clear negative slope. There is also a hint of a vertical line through the centre of mass. If so, it would represent some fairly rare occurrence for which we have not controlled. An important value of a partial residual plot is in examining a lesser effect after controlling for a greater effect, as in Fig. 7, examining dependence on memory. Here we can see not just a fuzziness induced by the other stronger variables, but a zero slope to an elongated cluster of points, with some outliers near the mean, again.

Tables 3 and 4 present summary coefficients from the fifteen weekly regressions for the cpu and i/o test probes. Since variability of the coefficients over time exceeded the formal standard errors of coefficients for any one week, we suggest these summary coefficients as approximate values. The tables present the median coefficients, together with two measures of variation about the medians.

Table 3. Coefficient estimates from CPU probe

	Median	Lower quartile	Upper quartile	Minimum	Maximum
cpu (h/cpu h)	0.950	0.939	0.964	0.904	0.984
i/o (h/MB)	0.00080	0.00071	0.00116	0.00061	0.00149
Memory (h/MB h)	0.011	0.003	0.013	−0.001	0.019

Table 4. Coefficient estimates from I/O probe

	Median	Lower quartile	Upper quartile	Minimum	Maximum
cpu (h/cpu h)	0.26	0.19	0.30	0.12	0.43
i/o (h/MB)	0.0067	0.0053	0.0091	0.0040	0.0107
Memory (h/MB h)	0.024	0.001	0.060	−0.049	0.092

In addition to the information collected under normal operating conditions, we collected similar measurements against artificial backgrounds. The backgrounds were formed by running multiple copies of jobs similar to the test probes. Although congestion could not be generated to order, it generally increased with more jobs and could be approximately controlled. The synthetic backgrounds did not exceed the observed range under normal operations for cpu use or i/o use (although we were explicitly trying to stress the machine) but we did get a larger range of observations of memory use. We regard the information collected during the synthetic background experiments as supplementary to that collected with actual backgrounds. However, they did provide information on the role of memory. The data from such experiments can also be displayed by partial residual plots.

Table 5 summarizes the coefficients for the three kinds of probes based on synthetic backgrounds.

Table 5. Coefficient estimates from synthetic backgrounds

	cpu probe	i/o probe	Memory probe
User cpu (h/cpu h)	0.83±0.02	0.13±0.10	0.34±0.08
System cpu (h/cpu h)	0.88±0.03	0.62±0.10	0.29±0.11
Disk (h/MB)	0.00059 ±0.00027	0.0022 ±0.0016	0.0024 ±0.0017
Memory (h/MB h)	0.012± 0.003	0.040± 0.013	0.084± 0.013

The first column of Table 5 can be compared with the first column of Table 3. The two lower cpu coefficients in Table 5 may result from not combining the two effects, which for the cpu probe seem similar. The disk coefficient of Table 5 lies within the inter-quartile range of Table 3 and the median disk coefficient of Table 3 lies within one standard error of the coefficient in Table 5. Thus these are not significantly different. Likewise the memory coefficients are not significantly different. We have used the coefficients from Table 3 in the following sections.

In comparing column 2 of Table 5 with Table 4, we see that the synthetic background produced different results. The disk coefficient is smaller and the waiting time has been ascribed to the other system cpu instead. These are more collinear when measured in the synthetic background data, and we distrust this result. (The simple correlations of system cpu time and disk access counts are 0.69 and 0.56 in the synthetic background and in the daily background.) We have used the coefficients from Table 4 in the following sections.

We do not have data for a memory intensive probe in the daily background, because we did not expect the daily background to have sufficient range. Therefore, we have used the coefficients in the third column of Table 5 in the following sections.

4. LOAD VARIATION MEASUREMENT

A UNIX system utility, the *system accounting report* (sar), gathers information on cpu and disk activity rates. It reports total usage during each hour, and is a rich source of information on cyclical patterns of load variations in cpu and disk activity.

Figure 8 shows plots of hourly loads during the work week. The top panel shows cpu loads, the middle panel i/o loads, and the bottom panel memory loads. The data for cpu loads and i/o loads are from a standard system report detailing all usage. This is preferable to our own data which sample usage, but which had to be used to estimate memory usage. Within each plot the solid dots represent a central estimate from the data—medians for the top two panels which had four or less observations at each hour and 10% trimmed means for the bottom panel

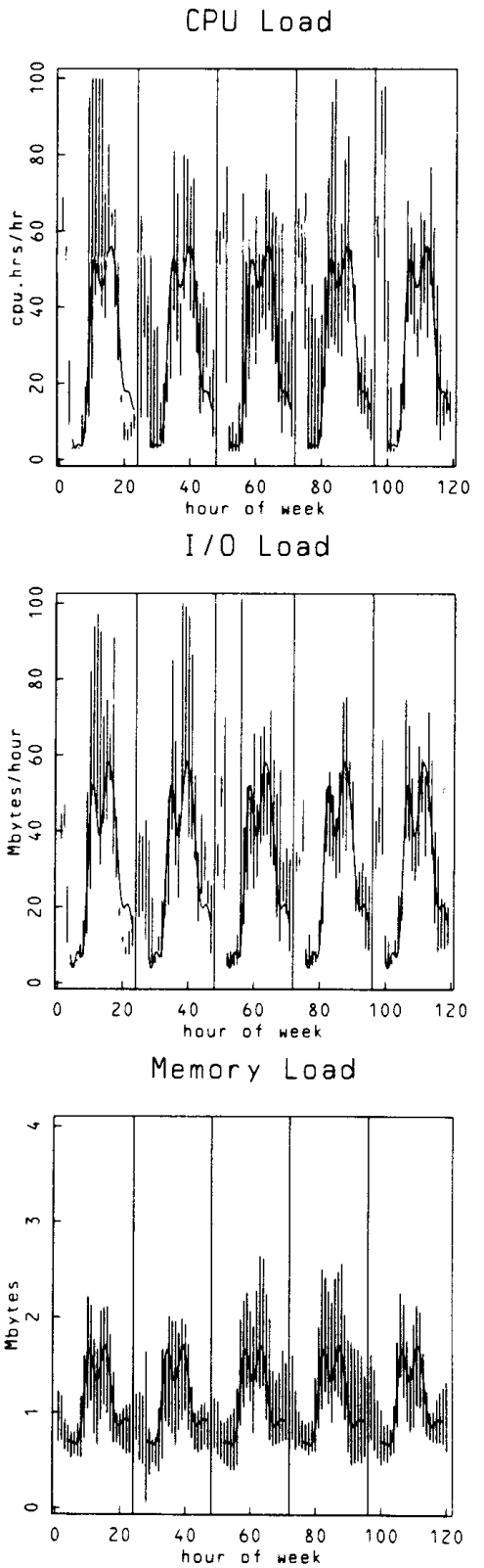

Figure 8. Weekly load cycle.

which had up to sixty observations at each hour. The vertical bars represent an indication of scale—ranges for the top two panels and 10% trimmed standard deviations in the bottom. The solid line represents a (24 hour period) Fourier fit to the solid dots in each panel. The coefficients were first fitted on a daily basis, but the coefficients were not significantly different by day. For instance, an F test of the hypothesis that the coefficients of the Fourier fit to the i/o load were the same day-by-day yielded 1.18 on 32 and 240 degrees of freedom. Therefore the Figures show the results of fitting over the period of a week.

Figure 8 illustrates several points. First, there is a systematic variation in load by hour of the day. This can be seen from the central points, and is highlighted by the amplitudes of the Fourier fits. Secondly, there is considerable variation from the central estimate at most hours—especially during the daytime. Thirdly, we do not see any need for distinguishing the weekdays. Each day seems fairly well fit by the 24 hour period Fourier fit.

A few other less consequential points need comment. A day's peak usage for cpu and i/o was sometimes in the early morning during the time of our experiments. This was the effort expended to implement the non-linear pricing system discussed below. It did not require large amounts of memory, however. We have dropped this period from our Fourier fit and from our calculations of marginal congestion prices below. The memory load does not drop very near to zero late at night, which suggests that we have included memory used by system processes. We deliberately excluded the space used by the operating system itself, however.

A final point is that the usage patterns are quite similar for each resource. The day usage is considerably higher than the night, with the afternoon usage higher than the morning, and with a noon-hour drop. The evening usage exceeds the late night usage.

The variation shown by this Figure, and apparently typical of computer loads elsewhere, has led some to suggest dynamic pricing mechanisms that will charge low prices when the actual load is low or high prices where the actual load is high. See Refs 11 and 12 for examples of such suggestions. In discussing this possibility with the system administrators, we were convinced that such a pricing system would have considerable administrative costs. A key point is accountability—the need to have verifiable records when a customer complains of a misbilling. (We know as users that suspected misbilling is not uncommon.) The necessity of maintaining a separate database on system usage levels to interpret prices charged would lead to greater difficulty in keeping sufficient records for accountability. Another issue is whether the non-cyclical variation is predictable. If it is purely random, then dynamic schemes basing prices for [now $+\varepsilon$] on the load at [now] would not be any better than using our Fourier predictions. Accordingly we do not suggest instantaneously varying prices, but prices which account for the average load variation by hour of the day.

5. DERIVING A PRICING POLICY

This section brings together the pricing rules of Section 2 and the empirical results of Sections 3 and 4.

In Section 3.3 we presented estimates of a waiting-time (or thruput) function based on VAX-UNIX performance at Bell Labs, and in Section 4 we presented estimates of a smoothed form of the daily load cycle. Thus we may compute idealized marginal cost pricing policies based on the current load, if we are willing to choose a value for waiting time. This is doubtless a controversial choice, but for purposes of illustration we have chosen $v = \$50/\text{h}$. Other choices may be accommodated by linearly rescaling the vertical axis in our subsequent figures.

In Fig. 9 we illustrate the 'marginal congestion prices' which are implied by the marginal cost pricing rule (A.3) with current loads, our estimated thruput function, and this hypothetical value of time. The solid line in each panel represents the time-varying price of a particular resource: cpu in $ per cpu hour, i/o in $ per megabyte transfer, and memory in $ per megabyte hour. Each price can be decomposed, see Eqn (A.3), into three components attributed to cpu, i/o, and memory loads. These components are illustrated in Fig. 9 by dotted, dashed, and dot-dash lines, respectively. Note that cpu load is the largest effect for both cpu and memory jobs, for cpu it is the predominant effect. I/O activity is the most important determinant of the i/o price. The shape of the price cycle imitates the load cycle illustrated in the previous section, but the peaks are accentuated owing to the non-linearity in the congestion function.

We must emphasize that these prices are designed for current load patterns. Their implementation would doubtless have a serious effect on those patterns whereupon these prices would no longer be appropriate. Nevertheless, these prices do provide valid *directions* for pricing policy changes. A movement in the direction of these marginal congestion prices is essentially a gradient step in an iterative pricing scheme. We believe as a practical matter, that theories of demand response sufficiently accurate to predict user response to pricing changes are not likely to be available. Therefore, it will be impossible to know exactly how much to move in the direction indicated by current load prices. A simple model developed in more detail elsewhere suggests that moving half way to a set of prices based on current loads would not be unreasonable.[3]

In Fig. 10 we compare our cpu and memory marginal congestion prices with the study period prices for the system under study. The marginal congestion prices are linear in cpu and memory so if we plot the price of a cpu second as function of memory size we obtain a simple affine function with a positive intecept representing the pure cpu effect. We plot this price function in Fig. 10 for 4 times-of-day: 3 p.m.—the system peak, noon—the midday dip, 7 p.m.—the evening dip, and 7 a.m.—the morning trough. Because the memory price is small, these functions appear to be flat, even though they do in fact have positive slopes. In contrast the price in use was non-linear in memory use: linear up to 64 Kbytes and proportional to the square root of memory use after 64 Kbytes. There was a 60% discount for usage in non-working hours; we illustrate both day and non-day rates in the Figure.

Several important differences between prices used and marginal congestion prices emerge from the Figure. The marginal congestion price of memory is very low and therefore the marginal congestion price of small jobs tends to be much higher than their charged price and

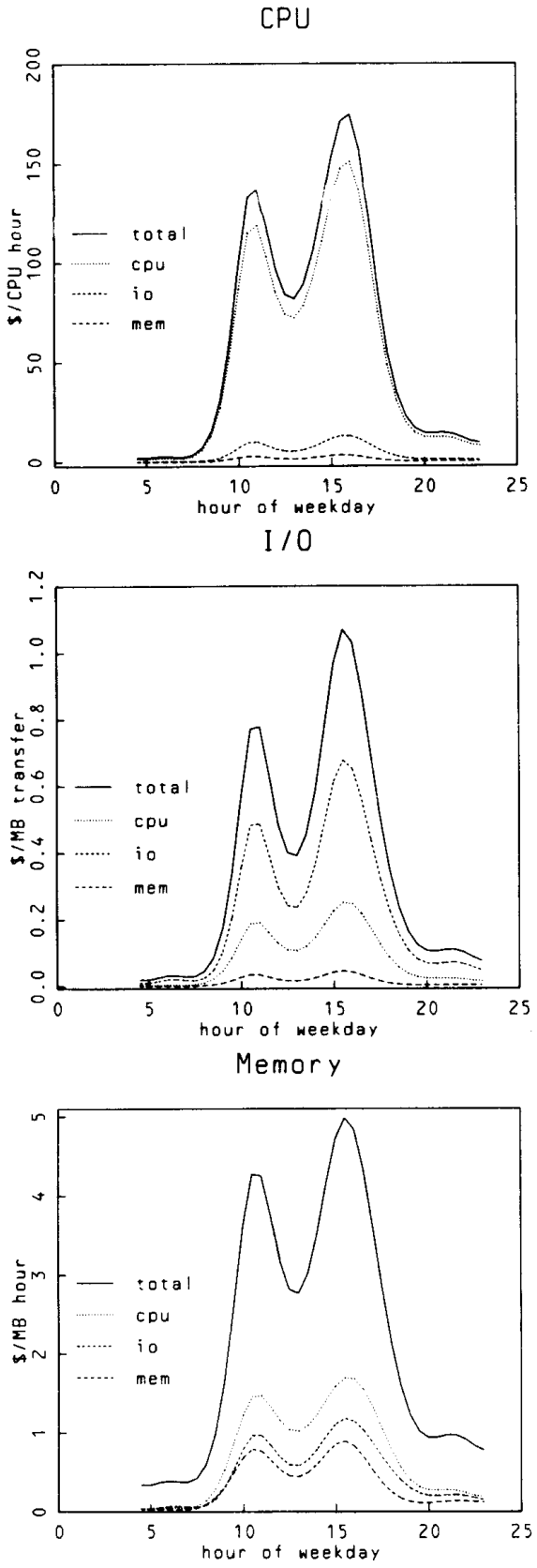

Figure 9. Decomposition of marginal congestion rates.

large jobs tend to be cheaper. This is in accordance with our earlier observation that current memory loads exert negligible impact on response time. The daily range of the marginal congestion prices was much greater than the range of charged prices. At the peak the marginal

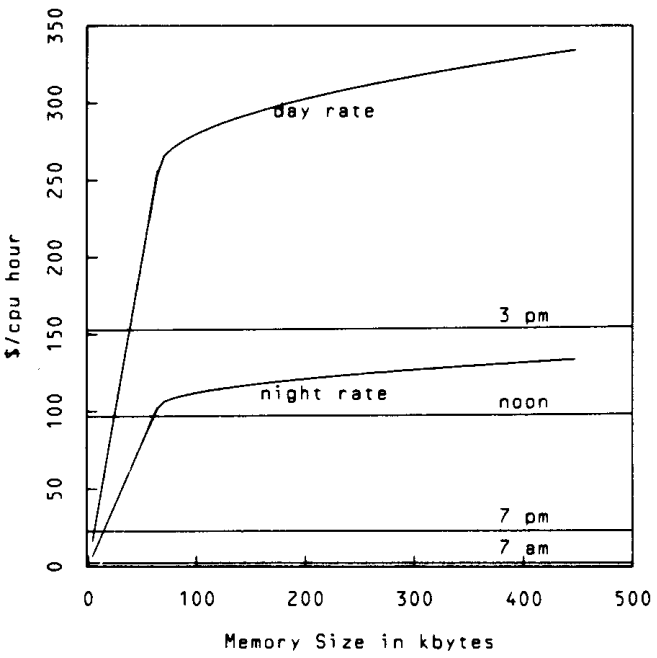

Figure 10. Cpu and memory marginal congestion prices.

congestion price of any job less than about 50 Kbytes was higher than its charged price. In the early morning, jobs larger than about 10 Kbytes had lower marginal congestion prices than their charged price. The disparities between charged prices and marginal congestion prices could be quite large, suggesting, as we pointed out in Section 2, that serious over use of capacity may have occurred in peak periods and serious under use may occur in the evening and early morning. Recall that the marginal congestion price is an estimate of the tasks' *social marginal cost* imposed on other users in the form of delayed response.

Another practical point is that smoothly varying prices would be administratively impossible. Given the large range (a factor of 32) over which the marginal congestion prices need to vary, it seems reasonable to suggest a few price levels which differ from their neighbours by factors of two. That is, a level of price appropriate to daytime peak use would be selected. The next lower price would then be half that level, and in our case might apply in hours starting at 8 a.m., 12 noon, and 5 p.m. The next lower price would be a quarter of the peak level, etc. This will define a set of fixed time periods during which prices are constant. For each user a record would be needed of their use of each priced resource in each price period. This is comparable to records currently kept for administering prices.

We want to repeat, we are not suggesting dynamically varying prices (prices that vary depending on the load level at the time), and we are not suggesting instantaneously varying prices. Such pricing systems do not seem feasible now. A discussion of priority pricing in this framework may be found in Ref. 3.

6. CONCLUSIONS

We have argued above that prices are potentially valuable tools for controlling congestion, improving utilization, and informing investment decisions for interactive

computer systems. To achieve these ends, prices should reflect estimated marginal costs of providing computing services. The predominant component of these costs is the value of time lost by users due to the congestion delay.

We have estimated models of response time and the daily load cycle on a VAX 11/780 UNIX system at Bell Laboratories to show the feasibility of marginal congestion pricing. Based on these estimated models we have computed marginal congestion prices which would be appropriate if user response to prices were negligible. The calculated prices varied considerably from the prices in effect. This shows the considerable room for gains in efficiency and fairness available from adopting a marginal congestion approach to pricing.

Acknowledgements

We thank Debasis Mitra for his patience in discussing the theory of queueing models. We thank Herb Fischer, Steven Goldsmith, and Art Gross for their enthusiastic co-operation in making the measurements reported here while the system was in daily use.

REFERENCES

1. L. J. McKell, J. V. Hansen and L. E. Heitger, Charging for computing resources. *Computing Surveys* **11**, 105–120 (1979).
2. A. P. Lerner, *The Economics of Control*, MacMillan, New York (1944).
3. W. A. Gale and R. W. Koenker, Pricing interactive computer services: a rationale and some proposals for UNIX implementation. *Bell Laboratories Economic Discussion Paper #250* (1982).
4. A. C. Pigou, *The Economics of Welfare*, MacMillan, London (1920).
5. A. A. Walters, The theory and measurement of private and social cost of highway congestion. *Econometrica* **29**, 676–699 (1961).
6. P. Naor, The regulation of queue size by levying tolls. *Econometrica* **37**, 15–24 (1969).
7. M. Marchand, Priority pricing with application to time-shared computers. *AFIPS Conference Proceedings* **33**, 511–519 (1968).
8. T. W. Pao, The UNIX system activity package. *Bell Laboratories Memorandum for File 3644–800313.01 MF*, March 13 (1980).
9. J. R. Jackson, Networks of waiting lines. *Operations Research* **5**, 518–521 (1957).
10. W. A. Larsen and S. J. McCleary, The use of partial residual plots in regression analysis. *Technometrics* **14**, 781–790 (1972).
11. T. L. Shaftel and R. W. Zmud, Allocation of computer resources through flexible pricing. *The Computer Journal* **17**, 306–312 (1974).
12. H. Kameda and C. C. Gotlieb, A feedback-coupled resource allocation policy for multiprogrammed computer systems. *Acta Informatica* **8**, 341–357 (1977).

Received October 1982