MICROECONOMICS AND THE MARKET FOR COMPUTER SERVICES

by

Ira W. Cotton

Institute for Computer Sciences and Technology

National Bureau of Standards

Washington, D. C. 20234

Microeconomics has much to offer the computer services manager. This article reviews some of the traditional topics in microeconomics and shows how they can be applied to the market for computer services. The topics covered include supply, demand, costs and pricing. The most significant application of microeconomics is in setting prices -- so much so that microeconomics is frequently called "price theory." Accordingly, the thrust of the article is towards providing a sound framework for the pricing of computer services.

CR Categories: 2.41, 2.11

Keywords: Economics, Microeconomics, Pricing

Acknowledgements

The author is indebted to Dr. James Dei Rossi of the National Bureau of Standards and Dr. Dale Rasmussen of George Washington University for their careful review of the manuscript and the many constructive comments they offerred. Responsibility for the content is, of course, the author's alone.

This work was supported in part by the National Science Foundation, Grant AG-350.

INTRODUCTION

The computer services industry provides an excellent environment for the study of microeconomic principles. (SH69) Even a single computer installation may be viewed as an economic system in miniature wherein may be observed all the forces of supply and demand. In this paper we offer the results of such an observation, both for individual installations and for the industry as a whole. We follow the canonical approach to microeconomics, and consider in turn the topics of supply, demand, costs and pricing. The emphasis will be on relating microeconomic theory to the practical management of computer services.

Pricing is the key factor, and developing a rationale for setting prices is the objective of the entire discussion. Indeed, microeconomics is often referred to as simply "price theory." Too few computer center directors realize that they are operating in a marketplace. An understanding of supply and demand relationships as they exist in the computer services marketplace and an understanding of the economic effects of a pricing policy should lead to a more reasoned approach to setting prices. This will work to the advantage both of the computer center and its users.<SA71>

SUPPLY

The computer services industry is a subset of the entire computer industry. Under a definition suggested by Selwyn (SE71b), computer services includes all of the computer industry except hardware manufacture and maintenance. It includes service bureaus, time-sharing firms, consultants, software producers, and data bank organizations. It also includes in-house computer facilities, encompassing their operation, programming, systems analysis and systems management functions. In an economic sense, firms that operate their own computing systems are in effect suppliers of computing services, although they may limit the sale or provision of these services to themselves. At the present time, in-house computer facilities produce the overwhelming majority of computing services in this country. Service bureaus and time-sharing suppliers represent a very small fraction of all such services produced.

Types of Services

Basic computer services are provided by the execution of a program or predefined sequence of instructions on a hardware complex of a CPU, main memory and peripherals referred to as a computer system. The basic service is the action of this program on a set of data which is provided for the particular execution. A basic service

supplier will offer the use of the computer system for the time necessary for the particular program to be executed./1/

Such "raw computation" is not, however, the only service generally offered. Organizations engaged in the provision of computer services tend to be vertically integrated in that they supply computer time, application software, systems analysis, consulting, training and other services to users. As Selwyn (SE72b) points out, these services are characterized by significantly different production functions, thereby providing opportunities for specialized suppliers, operating on a scale different from that of the integrated supplier, to produce certain services more efficiently.

Economies of Scale

The production of raw computation has been shown to exhibit increasing economies of scale over the range of currently available machines. In the 1940's, Dr. Herbert Grosch asserted that the power of a computer system increased as the square of its cost. Although unpublished by Grosch at the time, this part of the computing profession's early oral tradition has become firmly entrenched in subsequent articles as "Grosch's Law." (SO66)

Grosch's Law has been empirically tested by a number of investigators, including Knight (KN68) and Solomon (SO66). Both of these studies found that the law generally held, although

results were more in conformity for processor-bound tasks than for I/O-bound tasks, reflecting the more rapid drop with increased size in the average costs of main memories and logic elements than in mass storage devices and communications. More recently, Hobbs (HOB71) has claimed that the law was more a reflection of the pricing policy of a major manufacturer than an inherent law of computer systems design. Basing his argument on a perceived change in the relative costs of different parts of a computer and communications system, Hobbs states:

> "To the extent that Grosch's Law could be considered a law, it has been limited by the Software Amendment of 1964 and the Integrated Circuit Amendment of 1967 and has been repealed by the LSI Act of 1970."

While Hobbs does not present any empirical evidence to support his assertations, the implication is clear. Grosch's Law is essentially a statement about central processing units, and gradually loses its validity as it is extended to the other components of a computer system.

The economies of scale evident for raw computation do not extend to other types of computer services. Solomon's study of personnel costs for commercial (non-governmental) installation (SO70) did reveal smaller average costs for larger installations, but this was because larger hardware installations were being supported,

not because there are economies of scale in personnel per se. Indeed, if anything the reverse is true. The effective span of control for programming managers is limited, resulting in larger managerial overhead for larger projects, and the complexity of a programming project is commonly acknowledged to grow (perhaps even exponentially) with size (NA64).

Product Differentiation

Selwyn (SE71b) has argued that computing services, taken as a whole, are relatively undifferentiated from one another since, providing that the relative scale of hardware is selected properly, and assuming intelligent system designs, the development of most types of computer applications can be accomplished with almost equal success on any general purpose computer. Thus, prior to the actual commitment of resources to software development, the application developer is relatively indifferent among the alternative hardware configurations that may be available to him. However, as Selwyn recognizes, the services of a general purpose computer become highly differentiated when they are provided in conjunction with access to a specific application program. Users with a heavy investment in not-easily-converted software are often locked-in to a specific system. Given this observation, it would seem desireable to restate the original argument to be that only raw computation is a relatively undifferentiated product. Furthermore, even raw computation may not even be so

undifferentiated as Selwyn believes. Computers are not, in general, compatible with one another, and brand loyalty does exist based on real or perceived differences or simply which system the buyer was trained to use.

The other types of services which have been considered along with raw computation -- software development, systems analysis, consulting, training and user services -- are already highly differentiated. Most buyers demand more than just raw computation, so that even if raw computation is completely undifferentiated, through the other services the supplier may establish for himself an oligopolistic (for example, offering the services of a particular operating system or compiler) or monopolistic (for example, offering proprietary applications software) position.

Economies of Integration

A production function is a statement about the relationship between the inputs used in production and thr resulting output(s). It describes a technological relationship: with a given technology, certain combinations of inputs will make possible a given level of output. Production functions usually apply to one activity or to a closely related group of activities. A firm that produces several different types of products is said to be an integrated supplier and is subject to all of the production

functions that apply to the individual products. If these are parallel and not directly related to one another, the firm is said to be horizontally integrated. If, on the other hand, all the products are related in that they represent intermediate stages of the production of some final good or service, then the firm is said to be vertically integrated. Most firms in the computer services area are vertically integrated, offering services comprised of raw computation, specialized application programs, contract programming, consulting, and user services.

The advantages of vertical integration to a supplier of computer services are the protection to the supply of factor inputs it affords, the internal demand for intermediate outputs it creates, and the economies which often result from control over the entire production process. So attractive are the advantages of vertical integration, that in the computer service field it has occurred by growth in both directions -- hardware manufacturers have integrated upwards by the creation of service bureau subsidiaries and service firms have integrated backwards into hardware manufacture (as University Computing Company did for its line of COPE remote batch terminals).

Against the advantages of integration must be weighed certain disadvantages. As Selwyn (SE71b) explained, each activity of an integrated firm is characterized by its own production function. Software production, for example, is more efficiently done by

smaller firms, in contrast to the economies of scale associated with raw computation. Each function is likewise characterized by a most efficient scale of production. Thus, a single integrated firm of any given size is not likely to be the most efficient size for the production of all its products. When this occurs, other firms, competing with the integrated supplier, may be able to produce similar goods or services more efficiently, and hence capture a large share of the market. Of course, even an inefficient component of an integrated firm may be protected from competition by the more efficient components.

Market Structure

As we have noted, the market supply pattern for computer services has been torwards integration. Dis-integration, where it occurred, was generally limited to the labor intensive portions of computer services -- principally software development (where a form of economy of scale results from the negligible marginal cost for additional copies of a program), but also consulting, facilities management, and training. The dis-integration of basic computer services was hampered by technological difficulties.

Recently this has changed. The development of computer networks has provided a marketplace for the widespread sale and distribution of basic computer services. (HER73,HOO72) Users with

nothing more than a terminal and access to the telephone may select from a large number of potential suppliers. Since geography is no longer of concern, the economies of scale which were observed to exist for basic services makes practical the dis-integration of such services. Thus, a large complex located somewhere may offer basic services to users anywhere in the country at prices lower than they can obtain locally. Since personal services are still important and are still most efficiently offered by local concerns, wholesaler-retailer relationships have developed. (GR072,ST72) Large computer centers act as wholesalers of basic services to local retailers, who resell them to users along with the other services necessary for their productive use. It appears that this will be the characteristic industry pattern for the future.

DEMAND

The demand for computer services is a derived demand. Computer services are not required for their own sake, but are used for accounting, inventory control, market forecasting -- in short, for all the myriad business problems to which the computer has been applied. As a derived demand, the demand for computer services on the whole could be expected to be somewhat inelastic./2/ While this may be true for existing applications (the automation of a particular function is rarely reversible), it does not appear to hold for new applications.<CH67> For new applications, the

reduction in the unit cost of computing (over time) which has been characteristic of the industry has been a major factor in promoting the continued development of these applications.

The cross elasticity of demand for services from different vendors varies according to the homogeneity of the particular service. Homogeneous or undifferentiated services such as raw computation as a rule have high cross elasticities, reflecting the easy substitutability of products from different vendors. (Perhaps it would be more correct to simply say that the market for undifferentialted computer services is highly competitive). The cross elasticity of highly differentiated services such as specific applications programs will be lower, perhaps even zero. (Thus, as has been explained, a firm can establish itself as a monopolist through the development of proprietary software.) Negative cross elasticities of demand between different types of services might also be expected; e.g., a decrease in the cost of raw computation could stimulate increased demand for programming or consultation services.

In contrast to many other industries where industry-wide demand characteristics are well known but the demand facing an individual firm is not, in the computer services industry the demand facing individual suppliers has been most thoroughly investigated. Many computer service suppliers face captive markets, so that the aspect of the demand facing the firm which has been most

intensively studied is the regular variation in demand which often occurs on daily, weekly and annual cycles./3/

Computer installations are frequently faced with wide cyclical variations in the quantities of service demanded by users. (J071) Typically, demand for service is greater during prime shifts than at night. Demand may be greater one day a week when a payroll program must be run, or at the end of a semester, when student projects must be finished. Variations in quantity demanded may be estimated by the length of service queues at different times in the cycle. The demand function itself may fluctuate, since the users may be drawn from different populations at different times in the cycle. A major objective of computer center managers should be to level out these fluctuations so as to make more efficient use of the system and reduce the disutility to users who cannot obtain service at times of peak loading. As will be demonstrated later, the price mechanism provides the means to accomplish this.

COSTS

The provision of computer services is characterized by a high ratio of fixed to variable costs. This is most true for the supplying of raw computation, since for the most part, machine rental accrues whether or not the system is running production jobs. For this reason, considerable attention has been devoted in

the literature to the equitable allocation of these fixed costs among the various users. (BA67, DI68, GRA72, HO069, KR72, SE70, ST68) We shall refer to this cost allocation as "billing", rather than "pricing", since pricing has other objectives which will be discussed later. For a modern computer system, the design of an equitable billing algorithm is not simple. (Other types of computer services such as contract programming and consulting present less of a problem since variable costs are a more significant portion of total costs, and fixed costs may be allocated as overhead in proportion to variable costs).

The earliest computers were operated in a sequential batch processing mode, wherein each job occupied the computer fully for the length of time necessary to run to completion. Accounting was simple, as each user could simply be charged according to elapsed or so-called "wall clock" time. Time-sharing and multi-programming changed this, since multiple jobs could occupy the computer simultaneously. The elapsed time for any given job was no longer a function only of that job, but was also a function of the job mix. Timing was not a problem, since most advanced operating systems could determine actual running time for each program. More serious was the fact that each job used a different set of machine facilities. Depending on the job mix, conflicts could occur, resulting in less than optimal use of the total computer system. Thus, a given system could take different times

to run a set of jobs, depending on the order in which they were loaded.

Despite this inherent variability, billing algorithms were insisted to conform to the principles of reproducibility (result in the same charges for the same job, no matter when run) and equitability (be a function only of the resources actually used by the job). (KR72) Additional suggested attributes of a billing algorithm were auditability, understandability, and demurrage (charging for resources which, though they may not be in active use, cannot be used by others -- for example, dedicated peripherals or memory space). (H069) There is no general solution which satisfies all these requirements. Most approaches have been to charge average costs which are determined from analysis of a past "typical" time period. This results in repeatability by using constant billing factors for all identifiable resources used (e.g., CPU time, memory space used, lines printed), and approximates equitability, since users are charged in proportion to resources actually used. (HET71, KR72) However, such an approach ignores fluctuations in true cost resulting from job mix idiosyncrasies, inevitably results in inequities as average factors for resources may change over time, and may fail to encourage efficient use of the hardware.

PRICING

Any economic system must solve the problem of how to use scarce resources. The price system is the vehicle by which economic units express their preferences in a market context. When these preferences are uniformly expressed in terms of price, the strategy of allocating resources to those willing to pay the highest price insures the maximization of total utility realized by the use of these resources.

It has been observed that computer services are among today's scarcer resources. (NI70) However, prices are not presently the dominant allocative mechanism for these services. Pricing has been used for a number of other objectives (OL71), and other mechanisms have been used to allocate resources (SM68a). In this section we review some of the uses to which pricing has been put, and some of the alternative mechanisms for the allocation of services. We conclude with an exposition of the proper relationship of pricing to the provision of computer services.

Pricing Objectives

It is well recognized that organizations operate according to many different objectives, be they stated explicitely or not. Naturally, the pricing policy of an organization will bear some relationship to the organization's objectives. Selwyn (SE71a) has

identified some of the different objectives, and indicated how they are expressed in pricing policy.

Profit Maximization

One approach to maximizing long term profits is to work torwards maximizing short term profits. Total short term profits are maximized by increasing production (and accepting a continually lower price, in accordance with normal supply-demand considerations) up to the point where marginal costs equal marginal revenue (see Figure 1). For firms in the computer service industry, the bulk of the costs in the short run are fixed, so that virtually all marginal revenue represents a contribution to profit. There is thus a strong motivation to establish prices so that all machine time is sold (this is the point in Figure 1 where the marginal cost curve becomes vertical, indicating that any increase in capacity in the short run is impossible). However, this is most frequently accomplished by setting the price according to the average demand.

For firms in the computer service business, adherence to this policy may be far from optimal in terms of the long term profit maximization objective. First, it ignores temporal variations in demand. This may result in all prime time being sold, but none at night. Second, the policy ignores the monopoly potential of specialized computer services. We have already discussed how a

firm may establish itself as a monopolist by differentiating its products. Third, by selling to capacity during the peak hours, the quality of service may become substantially degraded (particularly important for time-shared systems), which may result in the loss of customers dissatisfied with the service they are receiving (see Figure 2).

Market Penetration

A policy of increasing market penetration may accomplish more than mere short term profit maximization in achieving long term objectives./4/ By foregoing current profits, the firm may be able to capture a much larger share of the market than would be possible without this policy. Since demand is less elastic for established users than potential users, the firm may then be able to alter its policies with respect to short term profit and still retain a major portion of its customer base.

Tie-in With Other Services

As has been discussed, integration is presently typical of most firms in the computer services industry. The appropriate pricing policy must consider the impact on all of the firm's products, not just the one for which a price is being established. For example, a time-sharing firm may establish a very low charge for initialy connecting to its system in the hope of stimulating usage for which it can charge.

Optimal Use of Computer Resources

For any given computer installation there is an absolute limit to its capacity to offer service. However, this limit is rarely approached, due to imperfect matching of demand for use of the individual resources to their availability. A gross example might be the idle time occuring at off-peak hours. A more subtle example is provided by a system whose printer is saturated. A pricing policy which for the first example encourages off-peak utilization, and for the second example discourages excessive use of the printer, may dramatically increase the total throughput of the system.

Pricing Alternatives

Having determined the (set of) goal(s) of its pricing policy, management must then examine the tools available for the establishment of rate plans and policies.

Pricing for Cost Recovery

One alternative for a pricing policy is to estimate utilization over a given period and set prices so that they cover all costs of operation, including profit if a commercial installation. However, such a policy assumes demand to be perfectly inelastic and, as Smidt (SM68b) has shown, can often be self-defeating. The best example of this is provided by the case of the newly

installed computer system which has considerable excess capacity available which is expected to be gradually used up as demand increases. The cost per unit time of owning and operating the computer is fairly constant over its life and depends only slightly on the amount of work done. From a common sense point of view, it is clearly advisable to encourage users to make full use of the available capacity early in the life of the computer system when excess capacity exists, and to discourage usage (or encourage more efficient utilization) later, when usage approaches the capacity of the system./5/

However, if charges for the computer are determined by allocating its total cost over the total usage for a given time interval (usualy a year), the charges provide incentives that are exactly the opposite of what is desireable. When the computer is new, the fixed costs are allocated over a small volume of work, leading to a high cost per unit of work. When the computer is old and nearing capacity, approximately the same fixed costs are spread over a much larger volume of work, leading to a low cost per unit of work. Insofar as users respond to the costs charged, they tend to economize on the use of the computer in the early days when excess capacity is available and to be liberal in their use of it later on when capacity is being approached.

The only way out of this dilemma is to recognize that the price at any point in time need not bear any relation to the cost of

production at that time. If demand for a good is low, its price may well fall below average cost, but thereby eliciting greater utilization. So long as marginal costs are covered, such operations will make a contribution to profit. Unless price is permitted to fall below cost, the proper information about demand may never be obtained, and the allocation of resources can never adjust to the unprofitability of that good. Smidt (SM68b) and Neilsen (NI68) have recognized the shortcomings of average cost pricing, and advocate the use of "flexible" pricing schemes where the price is allowed to vary to adjust to demand at any given time so that the quantity sold will be close to the quantity available.

Pricing According to Value

The characteristic negative slope of an aggregate demand curve arises in part from the fact that the value of a product or service -- perceived or actual -- may vary substantially from one buyer to another, and in part from the decreasing marginal utility of additional quantities of the product or service to a single user. In order to sell a larger quantity, it is normally necessary to lower the price to all buyers, even those who would be willing to pay more than is being asked, and to charge the same price for all quantities sold to the same buyer.

Price discrimination is a technique by which groups of users are isolated and charged prices which are closer to the maximum price

which they as a group would be willing to pay. Price discrimination may be accomplished by segregating users into groups according to their demand schedules and charging the groups different prices, or by charging individual users different prices for successive quantities of the the same commodity./6/ As shown in Figure 3, this has the effect of increasing total profit to the vendor. The larger the number of individual segments that can be isolated, the more profitable the technique will be. The requirements for price discrimination are that it be possible (practical and legal) to segment the market and that users in low-cost segments not be able to resell services to users in higher-cost segments. The ideal may be achieved by selling each unit of service at auction, so that the maximum price possible is always obtained. Sutherland (SU68) described a bidding technique for computer time, though he intended it as an efficient allocation mechanism rather than as a means to maximize profit.

Selwyn (SE71a) discussed a number of bases for market segmentation applicable to the sale of computer services:

> Segmentation by type of customer - for example, by offering discounts to educational customers, who would not purchase services were they priced according to their value to commercial firms.
>
> Segmentation by type of application - for example, a software

supplier can price individual program products according to their value to users, rather than their cost of production. User isolation is obtained by definition, since they are using different products.

Segmentation by time of day - this has been suggested by a number of authors as a means of more evenly spreading the overall load on a computer system over the total time available.

Priority Mechanisms

Priority mechanisms have received wide attention in the literature on managerial and operations research research problems. In contrast, they have been virtually ignored by economists. One group of authors (SI68) suggest that the reason for this is that priorities are simply a surrogate set of prices that may in some instances work as well as a true price mechanism but will almost never be superior. For their part, operations analysts seem unaware that priorities are a form of pricing; thus Kleinrock (KL67) discusses "bribes" which are merely prices, and Greenberger (GRE66) tries to minimize the cost of delay, a cost which can never be known except in terms of the price users would pay to avoid the delay.

Two types of priority rules are recognized (SI68): one which determines the access pattern for a given set of users, and

another which offers incentives to potential users in determining their demands for computer time. The problem with the first class of rules is that an implicit assumption must be made about the value placed on computer time by each user. In general, users will not value time equally, nor consider waiting equally costly, so such rules will not allocate time so as to maximize total utility to users. The second set of rules often suffers from inflexibility in the face of changing user requirements, and may discourage efficient substitution of other resources for computer use.

In defense of priority mechanisms, it is recognized that they may serve to reduce the level of disutility that users cause each other through their presence in service queues -- a function attributed by Marchand (MA68) to "advisable" pricing mechanisms. Priority mechanisms are also inexpensive to administer.

The Dual Role of Price

The controversy regarding the proper function of price has centered around whether it is a mechanism for the recovery of costs (including profit) or for allocating resources. Singer, Kanter and Moore (SI68) are quite emphatic:

> "This point should be stressed: prices are a rationing device, not a mechanism for recovering cost."

On the other hand, as Oliver (OL71) recognizes:

> "It is a sad fact of life that pricing is generally the only way a center has to recover costs. Someone has to pay for the center."

How are these opposing views to be reconciled?

A possible reconciliation may be achieved by recognizing that price has a dual nature and satisfies dual objectives. Any pricing policy will serve as an allocation mechanism (but with varying efficiency). As Nielsen (NI70) observes, "if resource allocation is not done explicitely, it will be done implicitely; there is no such thing as 'no allocation'." The concern of those who insist that pricing be viewed purely as an allocation mechanism is that this allocation be optimized for some set of criteria such as total user utility or system throughput.

The main thrust of the criticism torwards the cost-recovery objective is that it often focuses on the short term to the detriment of the long term, is frequently inflexible in its implementation, and thus may lead to inefficient utilization. Such objections are well taken, but can be met by aiming to cover costs for a more appropriate period of time and by adjusting prices in response to both secular changes and cyclical fluctuations in demand.

It should be possible to establish a pricing policy which satisfies both objectives of price. The overall result of the policy must be to achieve some cost recovery objective (maximize profit for a comercial installation, recover actual costs for an internal corporate installation, limit losses to a budgeted amount for a university center) as well as allocate resources on an equitable basis. Such a flexible pricing scheme can serve to promote more efficient use of the hardware and may even result in greater total revenues if the center had not been saturated at all times.

Microeconomic theory offers no prescriptions guaranteeing that costs can be recovered for a particular product or service. What it does offer are tools with which to analyze the level of production necessary for all costs to be recovered. One such tool is the so-called "break-even" chart (figure 4).

Break-even analysis assumes that all costs can be represented as either fixed or variable costs (or some combination of these two types), and that all units are sold at the same price so that marginal revenue is the same from each./7/ The break-even chart graphically illustrates the level of production required -- at a given price -- for the excess of revenue over variable costs to equal fixed costs.

This analysis has two shortcomings. First, the break-even level

of production may exceed capacity. If this is the case, cost recovery is impossible at the given price. The immediate temptation is to raise the price -- but this gives rise to the second and more serious shortcoming: the analysis ignores supply-demand considerations.

Raising the price will indeed steepen the total revenue line -- but there is no guarantee that the quantity sold (at the new given price) will be as great as the break-even point. Indeed, even the quantity which could have been sold at the old price (had capacity permitted) might have fallen short of the break-even point. If the equilibrium between supply and demand yields a quantity less than that required to break even, cost recovery is truly hopeless.

The effect of raising or lowering price, of course, depends on the price elasticity of the particular product or service under consideration. For products with high elasticity, a small change in price results in a large change in quantity demanded. In this case, increasing the price will not aid in recovering costs. (If however, the break-even point is below capacity, lowering price may aid in cost recovery by substantially increasing the quantity sold). For products with low elasticity, raising prices may indeed aid in recovering costs. It is for this reason that firms seek to differentiate their products or establish monopoly positions for themselves. Differentiated products tend to have

lower price elasticity, enabling the firm to manipulate price more freely without wide variations in sales.

The discussion should now be sufficient to indicate how a firm should undertake the establishment of a pricing policy to satisfy the dual objectives of resource allocation and cost recovery. The firm must have some knowledge of its own cost functions and of the nature of the market in which it is dealing./8/ Any requirements for "normal" profits can be treated as an additional cost. Possible "excess" profits cannot be determined in advance. The firm can then examine its break-even point for several different levels of price. This analysis, in conjunction with the realities of capacity limitations, will permit the firm to rationally manipulate price to recover costs (including profit) and control the allocation of resources. Cost recovery and the possibility of earning excess profits will normally be acomplished through commodities with low price elasticity. Where elasticity is high, pricing will be more directed at controlling allocation and restricting usage. Hopefully, a greater understanding of the underlying economic principles by computer center managers will lead to policies which better satisfy both goals.

FOOTNOTES

1. Since the execution of the program does not consume or in any way harm the computer system (except for the infinitesimally small amount of aging of active electronic components) and since for any given configuration capacity is strictly limited, payment for the use of the computer system may be considered as a true rent in the economic sense. (Strictly speaking, if we consider that system capacity may grow over time through the addition of new equipment in response to high demand, then we would have to speak of a quasi-rent).

2. The elasticity of a functional relationship between two variables is defined as the ratio of the relative changes of the two variables when the independent variable is changed by a small amount. Expressed this way we have $e = (dQ/Q)/(dP/P)$. (It is often easier to rearrange terms to obtain $e = (dQ/dP)(P/Q)$. In this form, the elasticity is determined by taking the first derivative of the function (with respect to the independent variable) times the ratio of the independent to the dependent variable.) Elasticity is a measure of the sensitivity of the value of the dependent variable to a change in the independent variable. If the sensitivity is high (has a value greater than 1), the relationship is said to be elastic. If the sensitivity is low (has a value less than 1), the relationship is said to be inelastic.

3. In this case the cross elasticity of demand for service from different suppliers is zero and the focus is on the price elasticity of demand (which is not zero).

4. In economic terms, the "short term" is defined as that period of time for which productive capacity is fixed. In contrast, in the "long term" productive capacity may be altered, either positively or negatively.

5. This may also be viewed as a penetration strategy as previously discussed.

6. The latter is a technique commonly employed by public utilities in the pricing of such commodities as water, gas and electricity. It should be noted that a portion of the price differential for successive quantities represents a passing on to the consumer of economies of scale in supplying the commodity; the remainder represents the "discriminatory" price decrease offered in order to sell additional quantities. An example in the computer world would be the reduced rentals charged by equipment lessors for second and third shift operation.

7. For a break-even analysis to be done properly, all costs must be discounted at an appropriate over the life of the project.

8. If the firm's cost function and the demand function for services can both be expressed mathematically then an analytic solution is possible. For example, consider the case of a linear, downward-sloping demand curve and a linear cost function:

Demand functions of the type shown in fig. 1 which appear to express price as a function of quantity demanded may also be interpreted as expressing quantity demanded as a function of price. In this form the function may be expressed as

$$D = Q - kp \qquad (1)$$

where D is the quantity demanded,
Q is the y-intercept (quantity demanded at zero price),
k is the slope of the line, and
p is the unit price.

The cost function we consider is of the type shown in fig. 4 with both fixed and variable components. Assuming constant returns to scale (a linear variable cost component) this function may be expressed as

$$C = f + vD \qquad (2)$$

where C is the total cost,
f is the fixed cost,
v is the variable cost per unit, and
D is the quantity demanded.

Profit, which we wish to maximize, is the difference between revenue and cost. Revenue is the product of the quantity supplied and the price. Profit, therefore, is

$$P = pD - C \qquad (3)$$

where P is the profit,
p is the unit price,
D is the quantity demanded, and
C is the total cost.

Combining (1) and (2) into (3) we obtain

$$\begin{aligned} P &= p(Q-kp) - (f + v(Q-kp)) \\ &= kpp + p(Q+kv) - vQ - f \end{aligned} \qquad (4)$$

In order to maximize this expression, we set the first derivative equal to zero and solve for the price:

$$dP/dp = 2kp + Q + kv = 0 \qquad (5)$$

$$p = v/2 + Q/2k \qquad (6)$$

This simple analysis does not contain a capacity constraint. Such a constraint can be easily handled, however. The quantity demanded at the optimum price (p in equation 6) can be found from equation 1. If this quantity is in excess of the available capacity, then p is increased until the quantity demanded is reduced to exactly the quantity available.

BIBLIOGRAPHY

BA67 Bauer, Walter F. and Richard H. Hill. "Economics of time-shared computing systems." (In two parts) <Datamation>, November 1967, pp. 48-55, and December 1967, pp. 41-49.

CH67 Chow, Gregory C. "Technological change and the demand for computers." <American Economic Review>, 57:5 (December 1967), pp. 1117-1130.

DI68 Diamond, Daniel S. and Lee L. Selwyn. "Considerations for computer utility pricing policies." <ACM Nat. Conf.>, 1968, pp. 189-200.

GRA72 Grampp, F. T. "A computer center acounting system." FJCC 1971, pp. 105-114.

GRE66 Greenberger, Martin. "The priority problem and computer time sharing." <Management Science>, 12:11 (July 1966), pp. 888-906.

GRO72 Grobstein, David L. and Ronand P. Uhlig. "A wholesale retail concept for computer network management." 1972 FJCC, pp. 889-898.

HER73 Herzog, Bertram. "Organizational issues and the computer network market." <Compcon73>, pp. 11-14.

HET71 Heterick, R. C. "Resource allocation through computer service charging." Second Annual SIGCOSIM Symposium, October 26, 1971, pp. 6-13.

HOB71 Hobbs, L. C. "The rationale for smart terminals." <Computer>, November-December 1971, pp. 33-35.

HO069 Hootman, Joseph T. "The pricing dilemma." <Datamation>, August 1969, pp. 61-66.

HO072 Hootman, Joseph T. "The computer network as a marketplace." <Datamation>, 18:4, April 1972, pp. 43-46.

JO71 Jones, Thomas L. "Load management for short turnaround." Second Annual SIGCOSIM Symposium, October 26, 1971, pp. 25-31.

KL67 Kleinrock, Leonard. "Optimum bribing for queue position." <Operations Research>, 15:305, 1967.

KN63 Knight, Kenneth E. "A study of technological innovation - the evolution of digital computers." Doctoral dissertation, Carnegie Institute of Technology, November 1963.

KR72 Kreitzberg, Charles B. and Jesse H Webb. "An approach to job pricing in a multi-programming environment." FJCC 1972, pp. 115-122.

MA68 Marchand, M. "Priority pricing with application to time-shared computers." <FJCC>, 1968, pp. 511-519.

NA64 Nanus, Bert and Leonard Farr. "Some cost contributions to large-scale programs." <SJCC>, 1964, pp. 239-248.

NI68 Nielsen, Norman R. "Flexible pricing: an approach to the allocation of computer resources." <FJCC>, 1968, pp. 521-531.

NI70 Nielsen, Norman R. "The allocation of computer resources -- is pricing the answer?" <Comm. ACM> 13:8 (August, 1970), pp. 467-474.

OL71 Oliver, Paul. "Pricing and the allocation of computer resources." Second Annual SIGCOSIM Symposium, October 26, 1971, pp. 1-5.

SA71 Sass, C. Joseph. "Benefits of equitable charges for a batch computer." <Data Management>, 9:3 (March 1971), pp. 23-25.

SE70 Selwyn, Lee L. "Computer resource accounting in a time-shared environment." FJCC 1970, pp. 119-130.

SE71a Selwyn, Lee L. "Computer resource accounting and pricing." Second Annual SIGCOSIM Symposium, October 26, 1971, pp. 14-24.

SE71b Selwyn, Lee L. "Competition and structure in the computer service industry." Second Annual SIGCOSIM Symposium, October 26, 1971, pp. 48-56.

SH69 Sharpe, William F. <The economics of computers>. Columbia University Press: New York, 1969. (RAND Report R-463-PR)

SI68 Singer, N. M., Kanter, H. and A. Moore. "Prices and the allocation of computer time." <FJCC>, 1968, pp. 493-498.

SM68a Smidt, S. "The use of hard and soft money budgets, and prices to limit demand for centralized computer facility." <FJCC>, 1968, pp. 499-509.

SM68b Smidt, Seymour. "Flexible pricing of computer services." <Management Science>, 14:10 (June 1968), pp. 581-600.

SO66 Solomon, Martin B., Jr. "Economies of scale and the IBM System/360." <CACM>, June 1966, pp. 345-440.

SO70 Solomon, Martin B. "Economies of scale and computer personnel." <Datamation>, March 1970, pp. 107-110.

ST68 Stefferud, Einar A. "The environment of computer operating system scheduling: torwards an understanding." <AEDS Journal>, March 1968, pp. 135-141.

ST72 Stefferud, Einar A. "Management's role in networking." <Datamation>, 18:4 (April 1972), pp. 40-42.

SU68 Sutherland, Ivan E. "A futures market in computer time." <Comm. ACM> 11:6 (June 1968), p. 449.

Microeconomics and the Market for Computer Services

(J18385) 13-AUG-73 11:30; Title: Author(s): Ira W. Cotton/IWC;
Sub-Collections: NIC; Clerk: IWC;
Origin: <NBS-TIP>ECONOMICS-COTTON.NLS;3, 10-AUG-73 07:09 IWC ;

Repky to ADO's bug report: (18328,)

I received ADO's bug reports (18328,) last week. 1

? IN UPDATE FILE-- This bug has been fixed. 1a

QUERY BUG--It is true that if NIC/query is used from inside nLS, on return, viewspecs may be changed and you are not always returned to the file you were in before entering query. This is because query makes use of the NLS link stack which is limited to 5 files; this is the same stack used by NLS jump mechanisms. If many jumps are made in query, the "older" files on the stack will be replaced. This is clearly, if not a bug, at least a mistaken implementation. Because, however, the entire query system is currently being redesigned and rewritten (and will avoid this problem by saving away the NLS stacks), and because it is used from within NLS be a limited number of people, nothing will be done about the problem immediately. 1b

Please continue to report any bugs you may find in the future. 2

HGL 13-AUG-73 11:41 18386

Repky to ADO's bug report: (18328,)

(J18386) 13-AUG-73 11:41; Title: Author(s): Harvey G. Lehtman/HGL;
Distribution: /ADO BUGS; Sub-Collections: SRI-ARC BUGS; Clerk: HGL;

IWC 13-AUG-73 13:21 18387

Cotton: Cost-Benefit Analysis of Computer Graphics Systems

Here is the other paper I promised to send you. This one isnt as polished as the first one, so please take it with the appropriate grain of salt and dont distribute it. I will send the illustrations for both by mail. Your comments would be much approciated.

COST-BENEFIT ANALYSIS OF

COMPUTER GRAPHICS SYSTEMS

Ira W. Cotton

Business Administration 265

George Washington University

1. Introduction 1

Computer graphics is a sub-discipline within computer science which deals with the manipulation of digital representations of pictoral information, including output of the picture on an appropriate display device and modification of the image in response to operator-initiated actions. There are many systems based on this technology in use today, primarily in the area of computer-aided design, and many cost savings are attributed to such systems. Hard evidence to support such claims is lacking, however. 1a

Few studies have been published relating economic principles to the design and implementation of computer graphics systems. Those that have been published deal with only one side of what is essentially a two-faceted problem. Either a detailed analysis of the costs and performance of an optimized system is presented, without corresponding analysis of the economic benefits to be derived; or expected cost savings are estimated without discussion of the costs of a system to achieve those savings. 1b

What is required are analyses which link these two kinds of information. The first type of study may be characterized as a cost-effectiveness evaluation; the second as a benefit analysis. Cost-benefit analysis would describe the desired type of study which would essentially provide return on investment information. 1c

Some comments are in order regarding the differences between cost-effectiveness analysis and cost-benefit analysis. Cost-effectiveness studies are aimed at optimizing the performance of a system according to stated criteria for a given level of investment. The sensitivity of system performance to changes in level of investment may also be considered within the scope of cost-effectiveness analysis. 1d

Cost-benefit analysis, on the other hand, is concerned with providing adequate information for making the investment decision. Is the stream of benefits resulting for the use of a system with a given level of performance greater than the required investment? If performance and/or benefits do not vary linearly with cost, what is the optimal level of investment (to maximize benefits)? Such are the questions answered by cost-benefit analysis. 1e

Cost-effectiveness analysis is necessary but not sufficient for cost-benefit analysis. A valid cost-benefit study requires that the optimal level of performance for a given level of investment be considered. Alternatively stated, for each given level of investment, the value of the benefits considered should be the maximum possible. This is precisely the information provided by

cost-effectiveness studies. What cost-effectiveness studies fail to provide is any assessment of the value of the benefits. Such assessments are often difficult to perform, but they are crucial to the investment decision. 1f

We will be concerned in this report with the methodology of cost-benefit analysis. Methodologies for cost-effectiveness studies will be considered too because, as has just been explained, they are essential for valid cost-benefit studies. The relevant literature will be critically reviewed and suggestions offered where published accounts are lacking. No attempt is made here to actually perform such an analysis. 1g

2. Computer Graphics System Design Alternatives 2

Computer graphics systems of the type under analysis consist, in their simplest form, of an analog display device similar to an oscilloscope, a display processing unit which converts digital display commands to analog signals to drive the display, and a general purpose computer system which generated the digital commands in accordance with some application program. This system is illustrated in Figure 1. 2a

This basic system may grow in a number of ways.<ME68> More and more features may be added to the display processor until it comes to resemble a computer itself. This permits the display device to be located remotely from the main computer, connected by a communications link. Since the link is generally slower than the processing rate of the display, local storage is required to hold the commands sent from the central computer. However, in this type of a system we permit only primary storage; secondary or mass storage such as a disk or drum is not permitted. This type of configuration is called a satellite graphics system because the graphics processor is located at a distance from the central computer but is dependent on it. The general configuration is illustrated in Figure 2. 2b

Such a system offers a number of tradeoffs for the designer.<PO69> The speed of the communications link, the power of the display processor and the amount of local primary storage all may be increased at increased cost. Also, certain tasks may be performed either in the main computer or in the local processor. The decision as to which components to upgrade and where to perform processing tasks is not obvious. 2c

The situation can get even more complex. It is well within the capability of today's minicomputers to control multiple displays located remotely from the central computer. In such configurations, it is again desireable to separate the digital processing functions from the analog conversion. In addition, it will probably be necessary at this stage to permit the graphics processor to have a mass storage subsystem. All of this results in a true network configuration where multiple users share facilities at each end of the communications link. This is illustrated in Figure 3. 2d

The attachment of input devices has not been considered in order to simplify the problem. In general, input devices send their data to the closest processor. In the first case this would be the main computer. In the latter two cases input signals would be initially processed by the satellite computer. The degree of processing performed at the satellite in response to inputs,

however, is a design variable and may vary widely from system to system.<C072> 2e

For each of these types of systems there are three types of tradeoffs which can be made: hardware/hardware, hardware/software and software/software. Hardware/hardware tradeoffs refer to the possibility of diverting costs from one component to upgrade another. For example, mass storage capacity at a satellite might be reduced in order to pay for a faster data link, or vice versa. Hardware/software tradeoffs refer to the possibility of implementing certain functions either in hardware or software. Character generation is a good example of a function which can be provided either way. Software/software tradeoffs refer to the possibility of performing certain processing tasks either in the main computer or the satellite. Display generation, or the process of formatting display commands from a description of the picture in a data structure, is an example of a function which may be performed in either place. All of these types of tradeoffs challenge the system designer to come up with the optimum for a particular set of criteria. This is what the art of system design is all about, and efforts to express these tradeoffs analytically is what is meant by making system design into a science. 2f

3. Analysis of System Performance 3

There are essentially two basic approaches to evaluating the performance of a computer system: observation and analysis.<SM68> The system is either set various tasks (benchmarks) to perform and it is determined how well it performs them; or the system is analyzed in terms of specific parameters in order to derive a measure of performance. The tradeoff between these two approaches is between the detail and accuracy of the analysis and the cost of performing it. The first method has the appeal of direct measurement: the system is assessed as a whole. However, benchmarks are expensive to prepare and to run. Parametric analyses are less expensive to perform, but their results are less reliable since the effects of small errors in determining the parameters may be magnified when they are aggregated. Specific techniques of system analysis fall somewhere on a continuum between pure observation and pure parametric evaluation. 3a

The first serious attempt to provide a formal basis for comparing computers was a listing of the various characteristics of every computer published by Adams Associates. This listing has been called an Adams chart; it is still published by Keydata and typifies a good simple portrayal of the classical features of computers manufactured worldwide. In the graphics area, Adams Associates also published the <Computer Display Review> which listed the features of graphics equipment; it was continued by Keydata and is now published by GML Corporation. 3b

As pointed out in <J070>, the limitation of the Adams chart is the inference that performance can be meaningfully predicted from these classic features. It has been well established that the use of the raw speed parameters of clock speed, arithmetic speed, memory speed, word size or I/O rate can be misleading in predicting comparative performance between different systems. Over the years, many people have taken different combinations of these classic features and have used these combinations as figures of merit to infer performance measures. 3c

Any listing of simple features fails to incorporate the differences that each system designer included to improve his system. The power of each instruction set combined with each system's architecture is not indicated; the effectiveness of system software is not considered; and factors for evaluating special capabilities such as multiprogramming or multiprocessing are lacking. 3d

A major advance over the feature analysis approaches was the development by the Auerbach Corporation of a standard set of small problems which could be programmed by experts for the variety of

different machines and configurations under consideration. The results yielded the amount of memory required, the amount of code and similar detail. The imlication is that performance on these standard problems is indicative of the performance on actual user problems, and that small-problem performance is indicative of large-problem performance. These implications may or may not be valid. 3e

As Johnson observes <J070>, all instruction mixes suffer from the limitations that the actual mix in use may or may not correspond to that of the "standard" mix. Further, none of the instruction mixes yet devised provide a reliable indication of full load behavior of computing systems. This is an especially important shortcoming when attempting to predict the performance of multiprogramming or multiprocessing systems. The full load performance of such systems does not appear to be proportional to its small-problem performance. 3f

Nevertheless, simple instruction mixes have been widely used over the years because they are simple, easy to use, and offer the statistical comfort that if it is risky to deduce a performance measure from one simple parameter, then it is less risky to imply performance from a set of simple parameters. The application of this technique is reflected in the set of test patterns which was developed by Adams Associated for their <Computer Display Review>. These patterns were designed to be coded for each display system to be evaluated. One immediate result would be a comparison of the number of words necessary to code the pattern in its entirety. The results of running each pattern were the percentage of the pattern that could be displayed flicker free -- a measure of the capacity of the display for applications of which the test patterns were supposedly representative. The patterns used were the following: 3g

Alphanumeric Test Pattern - The alphanumeric test pattern is representative of applications requiring the display of tabular or free-form character data. Figure 4 illustrates this test pattern. 3g1

Weather Map Test Pattern - The weather map test pattern is representative of a class of applications including geographical or mathematical contour mapping as well as sheet metal styling. Figure 5 illustrates this test pattern. 3g2

Graph Test Pattern - Graphs and charts are very common displays, both in business and scientific applications. Figure 6 illustrates this test pattern. 3g3

Architectural Drawing Test Pattern - The architectural drawing

test pattern is representative of applications where a large quantity of unstructured data must be displayed with high precision. Such applications include circuit board layout and mechanical design. Figure 7 illustrates this test pattern. 3g4

Electronic Schematic Test Pattern - The electronic schematic test pattern is representative of applications where pictures are constructed hierarchically, using repeated instances of certain patterns or subpictures. Such applications include logical design and certain layout problems and mechanical analysis systems. Figure 8 illustrates this test pattern. 3g5

A more sophisticated version of the instruction mix approach to the anlysis of system performance uses "kernels." A kernel is a complete nucleus problem; meaningful kernel problems are selected according to the type of application of interest. The execution time of a set of such nucleus problems is assumed indicative of the system's execution of the whole application. When kernels are programmed by experts on their respective machines and the actual software of that system is used, the results can be good indicators of that small-problem behavior of that system. 3h

The only reliable and accurate measurement technique that has been developed to completely analyze system performance is to actually code and run complete programs selected from those that are to be run in the actual application. Such sample programs are called "benchmarks." Pitfalls exist: the benchmark chosen may not accurately represent the nature of the work to be performed. Combining the results of several benchmarks partially alleviates this criticism, but choosing the right combination of benchmarks is difficult. Still, as long as benchmarks are run in the large and small mixes expected in the actual operation, a good indication can be obtained of both the large and small problem behavior of the system. 3i

The real limitations of the benchmark approach are that it takes a large effort to write the sample programs and to prepare realistic data. The task is difficult enough for normal computer systems for which some libraries of programs written in standard compiler languages have been developed for benchmark purposes. It is nearly an impossible task for computer graphics systems where the programming systems for each are completely different. Also, the results of running benchmarks are only as valid as the benchmarks are representative of the real application mix. Graphics programs typically are composed of a number of different modules with radically different characteristics. Furthermore, an essential part of the graphics programs are the sequences of operator inputs which direct the execution paths of such programs. These sequences are difficult to model and expensive to actually

generate in a test situation. Thus, developing benchmarks for graphics systems which are truly representative of actual applications is difficult to do and the benchmarks are equally difficult to run. 3j

A final technique which will be discussed is simulation. Simulation cuts across the two general approaches of observation and analysis. To use this technique, a model is constructed of the system under consideration and the performance of this model is "observed" under the desired conditions. Since the model represents a simplification of the actual system, analysis of the system is required to insure that the model accurately represents all the salient features of the actual. When the model is run with the desired (simulated) workload, the result is analogous to observing the actual system. 3k

Of course, the results of simulation can be no better than the many assumptions that go into the construction of the model and into the design of the simulator. Constructing a computer system simulator for performance evaluation purposes is, as has been pointed out,<CA67> not an easy task. If the level of simulation is too fine, the simulator will be too costly to use. If the level is too gross, not enough information will be yielded. Despite these difficulties, simulation is becoming much more widely used, and a good simulator properly used may be one of the best tools for predicting system performance accurately. As we shall see presently, it is also a technique which can and has been used to analyze computer graphics systems. 3l

5. Cost Analysis 4

The problem of cost analysis is to express all the costs of alternative system proposals in a form suitable for analysis. This requires that ALL costs be included, and that all costs be expressed in a comparable form. The question of which costs to include has been well treated in the literature. Joslin <JO71> presents a detailed analysis of all the types of expenses which must be included in costing a system design alternative. The main complication in many cases is the need to include development costs in the analysis. 4a

If two or more development projects are to be compared, it is necessary to compare the amount of programming effort required in each case. Data for such an analysis is "softer" than data describing the pricing of components or even charges for the use of components. The problem is to accurately estimate development expenses in advance of incurring them and to estimate the number of systems which will benefit from that development. Substantial work has been done in developing formal techniques for estimating such costs <LA66, NE67>, but their main effect has been to reduce the margin of error from outrageous to large. Yet the assumptions made about the cost to implement functions in either the central site or the satellite and the number of replications over which to allocate this development cost may be dominant in the analysis. 4b

For example, large systems today all come with operating systems, and most with communications handlers as well. On the other hand, few minicomputers come with the type of operating systems which would be required to shift some functions from the central site to the satellite. The cost to implement such an operating system may be critical to determining the cost-effectiveness of the proposed division of labor. Of course, given a large enough number of satellite systems, the average development cost may be made arbitrarily low. Thus, an honest and realistic assessment of the expected number of replications must be made. 4c

The other problem of cost analysis is to express all costs in comparable forms. The assignment of unit costs to the use of the various resources is of critical importance. Ideally, all resources would be priced at their incremental costs. For resources which would not be present if there were no graphics system, incremental and average costs are the same. More important, if any tradeoff is to be valid, costs for resources in the central system and at the satellite must be comparable. This can be accomplished by using, in all cases, equivalent monthly rental charges or resource utilization charges, where appropriate. 4d

For example, a graphics processor cannot be utilized at no cost

simply because it is owned and always available. An equivalent monthly rental charge must be determined and used to derive unit-time operating charges so that they may be aggregated with the charges for the central processor, which are likely in this form already. Likewise, file storage charges for devices at either location must be represented in equivalent, non-zero terms. 4e

The concept of fixed versus variable costs is also a relevant one. Development costs such as design and programming costs are fixed, as are hardware components in the graphics terminal since they cannot be used for any other purpose. In contrast, the use of central site resources such as storage and the execution of programs are variable costs, since the central computer is shared by a large number of users who only pay for the units of each resource as they are used. Some charges may be either fixed or variable depending on the particular arrangements. For example, communications charges would be considered as fixed if a leased line were used and as variable if dial-up facilities were used. 4f

The significance of breaking out the charges in this manner is that the total cost for a given configuration during a given time interval, say a month, depends on the level of usage during that interval. Some authors such as Prince <PR71> have stressed the need for high utilization of the graphics equipment in order to lower the unit costs of operation by spreading the fixed costs over a larger number of operating units. This is a different matter. The sensitivity to level of usage may be reduced if fixed costs can be converted to variable costs. Certain design decisions offer this opportunity. 4g

For example, a higher speed communication link may make it possible to store programs and data at the central computer which are used by the graphics program being executed. If access to this storage is sufficiently rapid, it may be possible to reduce the amount of memory required at the satellite, which is a fixed expense. Unfortunately, the nature of the tradeoffs between fixed and variable costs is not always so clean cut. Frequently it is less expensive to provide for a function as a fixed only above a certain level of utilization. Communications charges are an excellent example -- above a certain level of utilization it is cheaper to lease a full time link than to make a call each time. So estimates of the level of utilization for a system may still be critical to system design decisions. 4h

Generally, the problems of cost analysis for computer systems are well-understood. Computer graphics systems are not so unique in this regard that they present any special problems. Perhaps the greatest challenge for cost analysis is to obtain results in a form such that the marginal costs of performance were obvious or

could be easily determined. This would permit the costs for improved performance or the savings from reduced performance to be more easily used in making design decisions. A parametric formulation explicitely relating costs to level of performance could serve as input to a decision model. 4i

Cost-Effectiveness 5

Many studies have been published discussing the tradeoffs involved in the design of graphics systems. <JA71, MY68, PO69, TH67, WA67> However, few published results show any serious attempt to quantify these tradeoffs. Quantification is necessary if the results are to be handled in any analytic way. One of the first published results of a quantitative analysis of design tradeoffs was Foley's study of satellite graphics systems.<FO71> 5a

Foley defined optimum design of a computer graphics system as "maximizing" a display system's performance subject to a cost constraint." In his view, optimum display design can be thought of as a resource allocation problem. The resources are dollars which can be allocated to the purchase of display subsystems of differing individual performance. System response time was employed as the sole measure of system performance. (Minimizing response time means maximizing performance). Total system performance is determined from a model of how the subsystems fit together. The parameters of the model are functions of the capabilities of the graphic hardware and of the computational requirements of the graphics application. The model can be analyzed using numerical queueing analysis or simulation to obtain an average response time prediction. By applying an optimization procedure, the "best" graphics system configuration, subject to a cost constraint, may be found for several applications. The optimum configurations are in turn used to find general system design guidelines. 5b

Foley used a combination of kernels and instruction mix techniques to evaluate the computer power of the remote display controller. Basically a set of "display commands" was defined. These commands are at a level higher than machine instructions but lower than a high level language. The commands are coded in the machine language of the controller and timed. Each display command timing is weighted by a factor representative of the relative frequency of execution of that command for a particular application. The greater the value of the weighted instruction execution rate (the reciprocal of the command execution timing) the better suited is the corresponding controller for the application. The full account of Foley's research <FO69> indicates that he also employed the Adams' test patterns in a similar manner to compare the suitability of display controllers for different applications. 5c

Foley's analyses yielded curves of the form shown in Figure 9. Such curves show the price which must be paid for a given level of responsiveness. Performance is actually the reciprocal of response time, so that figure 9 illustrates the relation between performance (output) and cost (input). The decreasing slope of

the curve indicates the decreasing marginal productivity of additional units of cost in increasing performance. The analyses also revealed that different regions on the curve were associated with the upgrading of different components in the system. Simply stated, this means that different components were being upgraded when performance was improving most rapidly than when performance was increasing less rapidly. Thus the analysis yielded a set of decision rules indicating the order in which a limited amount of additional money should be spent. 5d

Cislo expanded on Foley's approach by considering a more complex model of a graphics system in greater detail.<CI72> Foley neglected the cost of resources at the central computer; Cislo's study includes them. Cislo also models a multiterminal system in which each terminal is used for a different application; this goes beyond Foley's analysis. Finally, Cislo employed simulation in contrast to Foley's use of queueing analysis. 5e

Cislo's model of a graphic system decomposed the operation of the system into a number of low level operations which might be viewed as the kernel problems of a display system. The activities included in the model are listed in Table 1; Figure 10 illustrates the model itself. A GPSS program based on the model was written and used to exercise it. The program simulates operation of the system by generating transactions which correspond to user requests. The interarrival time between requests is determined randomly from a negative exponential distribution. Similarly, the selection of the appropriate routines to process the request is determined stochastically. 5f

5f1

Activity	Mnemonic
Attention Processing	ATNPR
Transformation	TRANS
Text Processing	TXTPR
Tracking	TRACK
Application Processing	ASUBR
Data Transmission to Host	DTRTH
Host Computer Processing	HOST
Data Transmission from Host	DTRFH
Data Conversion	DCONV
Secondary Storage Operation	STORG
Update Data Structure	UPDTE
Graphic Order Generation	GENGO

Table 1. Model Activities

5f1a

The parameters of the model are very similar to those of Foley's model and the data were similarly obtained. Published data were used for simple parameters such as instruction execution and data transmission rates, and simple test programs were generated for the estimation of such parameters as the time required to generate graphic orders. Application-dependent parameters such as the probability of generating a given number of graphic orders were much more difficult to estimate, and in many cases are little more than guesses. 5g

In exercising the model, it was decided to hold these latter parameters constant while varying the hardware parameters. Accordingly, the operation of eight different hardware configurations, involving changes to the main CPU, display CPU and data transmission facility, was simulated. The results were in accordance with Foley's findings, but the method accorded much more detailed descriptions of system activity. The expanded version of the model which would handle multiple independent graphics consoles was described but not implemented. 5h

These techniques which have just been described offer much hope of permitting the analysis of computer graphics to be performed on a much more systematic and quantitative basis. The value of the analysis, however, is dependent on the validity of the variable(s) which is optimized as an indicator of total system performance. Accordingly, this discussion of cost-effectiveness is concluded with a discussion of this question. 5i

The response time provided to a user has long been considered a critical parameter in the performance of interactive systems. The general approach of system designers to this parameter has either been to "minimize it", subject to available monetary resources, or to insist on a minimum response time, e.g., 2 second response 95% of the time, regardless of the cost. This approach is overly simplistic and may be far from cost-effective for a number of reasons. 5j

First of all, not all interactive tasks require the same responsiveness of the computer. Miller <MI68> has shown that there are several classes of interactive activity with quite different response requirements. One class is the input of data by various means. An immediate response of no longer than .1 - .2 seconds is required for this class to signify acceptance of the data. A second class is characterized by a user engaged in high-intensity "brainstorming" requiring the ready access of data from his own "short-term memory." Such activity requires no longer than a two-second response in oder that the chain of thought not be broken. A final class includes those activities which complete a subjective (sub)task or (sub)purpose. More

extended delays (up to 15 seconds or more) may be permitted following such an activity completion, or "closure", than in the process of obtaining a closure. 5k

These findings may have an important impact on system design, since alternative methods exist to provide each type of response. For example, verification feedback and simple data manipulations may be supported on a so-called "intelligent terminal" with its own local mini-computer. Since the response from complex application-dependent computations may be permitted to be less rapid, these may be provided by a large time-shared computer, possibly located at some remote location and connected by a communications circuit. A cost-effective design will assign the function of providing each type of response to the facility which can meet the required resposne and meet it at least cost. 5l

Other factors may also be considered relating to response time. A uniform response time has come to be recognized as very important (in contrast to a response time which varies widely from transaction to transaction) -- so much so that in some applications the responsiveness of the system is artificially delayed when it falls below the desired value, so that the variability of response is thereby reduced (at the expense of lower average response time). 5m

Finally, some experimentation has been performed into the effects of restricting the free access to the computer after it has responded to each major request (similar to a closure).<BO71> It was found that "users tend to become dissatisfied if mild restraint is placed on their free interaction with the computer." However, "they also tend to problem solve more effectively, using less computer time and less of their own time in the process." The authors suggest that "the results cast doubts on the validity of user acceptance as a general index of system effectiveness." 5n

Clearly, the matter of the responsiveness required of a graphics system is not a straightforward question, and it does require serious consideration on the part of the system designer. The dilemma is exacerbated by the demonstrated sensitivity of responsiveness to additional investment.<FO71> 5o

Another effect of the over-reliance by analysts on response time as the sole measure of performance has been the neglect of a number of other important performance characteristics. These have been conveniently divided into task-dependent factors and human-dependent factors.<AU68> Task-dependent factors include screen size, message size, message format, erasability, color and half-tone capability in addition to response time. These factors will vary with importance according to the particular application;

hence a weighting approach should be used when evaluating systems according to these factors. 5p

Human-dependent factors, or more simply human factors, permit the system to be evaluated on its usability by human beings. Such factors include brightness, contrast, resolution, readability and the visual fidelity of the display. These factors are not as importance as they were in the early days of display system design when they were less-well understood, but they should still be validated for any system under consideration. Other ergonomic considerations might also be important in special cases, for example if the display were meant for group viewing. Again, the needs of the particular application will dictate requirements. A table in <AU68> presents a cross reference of desireable characteristics for specific display applications. Both task-dependent and human-dependent factors are included. 5q

6. Benefit Analysis 6

Methodologies for performing benefit analysis are analogous to the methodologies for evaluating system performance. There are two general approaches: empirical test (benchmarks) and analysis in terms of parameters. 6a

An experiment to determine the benefits of a given system would require two control groups and a typical task. One group would perform the task according to the old or baseline method; the other group would perform the task with the new system. Two measures of benefits could be obtained from such an experiment: cost savings from using the new system to achieve the same level of performance as the old method, and benefits from any performance levels achieved in excess of what was normally accomplished the old way. 6b

The other approach of assessing benefits by parametric analysis requires that tasks be decomposed into a number of elements. The benefits from improved performance in each element must be assessed, and then the old and new system can be compared on each element. For each application, the required tasks can be reconstructed from an appropriate aggregation of elements, and in this way the benefits from doing the particular application with the new system can be estimated. 6c

Unfortunately, very few published results of either type of study are extent in the literature. Productivity claims of 500% and above for the first type of study have been bandied about at professional society meetings, but the supporting evidence has not been published, either because it did not exist in satisfactory form or because such data was considered proprietary. What is most commonly found in this class are simple comparisons of an installed system with the old way of doing things, without controls for any other variables. Occasional studies of the second type have been published, but the focus has generally been so narrow that the results could not be generalized.<CA70> An example of a better study of this type is Gold's study of problem-solving performance in time-sharing versus batch processing.<G069> 6d

Using a programming language available in both batch and on-line environments, Gold "focused on a development of a methodology through which time shared computer system usage could be evaluated." Five categories of variables were included in the methodology: 6e

Cost of using the system
System performance

Turnaround time
The mode of learning resultant from system use
The attitudes of the system users 6e1

Principal variables associated with the measurement of the computer system features included: 6f

Varied computer response delays
Different degrees of interaction between the user and the computer system
Qualitatively varying response in feedback mechanisms and programming systems 6f1

User attitudes are defined in the study as "the degree to which the computer system characteristics appear to the user to facilitate or hinder him in the attainment of his short and long term objectives." User behavior is further characterized in the study as "the degree to which the user relegates programmable problem solving to the computer system." Performance for the purposes of the study is defined as "a measure of the output of the man-computer system which is arrived at independently of the user's behavior or the computer system used." 6g

The following are the significant findings of the study with respect to user behavior: 6h

The users of the time sharing system interacted with their computer system more than three times as often as did the batch-processing users. 6h1

There was no significant difference between the reasons advanced by the time sharing or batch-processing users for initiating computer actions. 6h2

There was a strong relationship between a batch-processing user's performance level and the number of computer interactions which produced usable output. For the time sharing user, the correlation between performance and the number of sessions with the time shared computer console is strongest. 6h3

Much more favorable attitudes toward the time shared computer system, its use and the results produced through it were evidenced. 6h4

With respect to quality and cost of results, the following results were reported: 6i

Use of the time sharing system resulted in a higher level of

objective performance than use of the batch processing system. Also, the "perception and understanding of the problem" of the time sharing users was evaluated as significantly higher than that of the batch users. 6i1

The total cost of time sharing and batch usage did not appear to differ appreciably. Higher computer costs and lower man costs were found with time sharing usage, however. There were indications that the total cost for equivalent objective performance would have been less for the time sharing users. 6i2

Application-oriented functions are different in character than the type of model activities developed by Cislo. For example, text editing and geometric sketching might be two tasks elements which would be used in different proportions in a particular graphics application. Determing the benefits of improved performance on each of these tasks would provide the necessary data to determine the benefits for an application comprised solely of those two activities. A sample list of kernel activities for benefit analysis is presented in Table 2. Other elements need to be identified and analyzed. 6j

Text Editing - original input and modification
Sketching - original input of geometric data
Updating - modification of geometric data
Inquiry - transaction processing
Directing - on-line control of other operations
Monitoring - of other operations
Mechanization of Output - production of finished drawings
Rapid Turnaround - should be isolated as a separate benefit

Table 2. Kernel Graphic Tasks for Benefit Analysis 6j1

The paradigm suggested would employ first analysis, then experimentation, then synthesis and finally validation by means of experimentation again. The goal is to isolate the individual task activities in an application, determine the individual benefits for each, aggregate these benefits to determine the benefits for the entire application and finally to validate these results by determining directly the benefits for the entire application. Since part of the paradigm includes experimentation using the application in its whole or total form, an application of limited complexity should be chosen. 6k

Having chosen such an application, it will be necessary to try to decompose it into discrete task areas. This is a procedure for

which no formal rules exist. Common sense and some understanding of what constitute "closures"<MI68> in the psychological sense will have to guide this analysis. The activities should be large enough that they may be analyzed and the benefits from each determined, but they should also be sufficiently limited in scope that they may be experimentally manipulated reasonably well and that they may be used as building blocks to characterize full applications. 6l

Having chosen the task areas, or kernels, experiments will be designed after Gold's model to determine the benefits derived from performing each task on a model graphics system as compared with doing it some other (baseline) way. Experimental procedures employing two or more sample groups and a control group are well understood and should be followed. Statistical techniques such as analysis of variance may be useful both in establishing the level of benefits and validating the initial choice of the groups. 6m

The final validation of the approach, however, can only come after an application is modeled by the task groups and the benefits are predicted and verified by experimentation. This will require an experiment which treats the entire application -- which we have been trying to avoid -- but this is necessary a few times in order to validate the approach. If the approach can be validated, experiments on that scale will not be necessary again. 6n

Studies such as this are but a first step in the development of kernel problems for the assessment of benefit values for particular systems. Many more studies, and preferably a systematic project with the stated objective of disaggregating the application-oriented functions of a graphics system and determining the benefits of each, are required before an analytic approach to determining the total benefits of an arbitrary system can be developed. 6o

7. Synthesis: Cost-Benefit Analysis 7

Cost-effectiveness analysis is valuable because it shows how to best allocate limited resources so as to optimize some specific performance objectives. However, such an analysis is inadequate for making an investment decision. Such decisions require that the absolute level of total system cost be compared with the value of the benefits provided by the system. This requires that acceptable levels of performance be defined, so that a point on the cost-performance curve can be selected to use in determining total system cost. 7a

In some cases the benefits for a particular application may vary with the performance of the system. This requires the consideration of a set of equations and the use of calculus in order to obtain an optimum cost solution. Expressed mathematically: 7b

$$P = f1(\$) \quad (1)$$
$$B = f2(P) = f2[f1(\$)) = f3(\$) \quad (2)$$
$$NB = B - \$ = f3(\$) - \$ = f4(\$) \quad (3)$$ 7b1

(Performance is a function of cost; benefits are a function of performance and therefore of cost also; it is desired to maximize net benefits which is the difference between benefits and cost). 7b2

The desired maximum occurs when the first derivative of the objective function is zero: 7b3

$$0 = d(NB)/d\$ = d[f4(\$))/d\$ \quad (4)$$ 7b4

This is the correct formulation of the cost-benefit evaluation problem. Unfortunately, it is rarely expressed in these terms in studies purporting to be cost-benefit analyses. 7c

One approach which has been described in the literature is the so-called "cost-value technique."<JO64> This approach describes a formal technique for comparing computer system proposals which all offer slightly different features beyond the basic mandatory requirements. The procedure involves assigning a value to each of these "extras" (expressed in terms of dollars) which are credited to the cost of each proposal. The value of expansion potential, for example, is assessed and subtracted from the costs of systems which offer it. The approach is crude and subject to all the peversions of subjective judgement, but at least it forces the evaluators to think in terms of the benefits to be gained from particular features. 7d

In another vein, Dunn <DU73> has sought to define a figure of merit for computer graphics systems which was free of bias towards any particular type of configuration but which is applicable to all types of configurations. However, Dunn claimed that "useful measures for interactive graphics systems are dependent upon subjective judgements." His argument was as follows: 7e

> "Any measure that is devised must reflect more than dollar costs and/or usefulness to the human and/or satisfaction achieved via a particular configuration. An interactive graphics system should and does function as an 'amplifier' of the interactive human activities in conjunction with the use of computers. The amplifying function affects quality of results, quantity of work accomplished, cost-effectiveness of methodology, efficiency of effort, minimization of elapsed time for work efforts, and so on. Amplification in this sense, then, incorporates a variety of dollar costs along with usefulness and satisfaction for the human user." 7e1

He suggested a figure of merit called "degree of amplification" or "amplification factor", symbolized as "A " and based upon four factors: productivity, degree of interaction, extent of graphics capability and total system direct dollar costs per graphics console. 7f

The productivity factor, P, is defined as the change in the rate of units of acceptable output relative to the same function being performed via a non-graphics console. The units of acceptable output are application-dependent (e.g., engineering drawings completed). 7g

Graphics capability is a measurement factor that reflects both ease of use of the graphics console and extent of load on supporting computing systems. The graphics capability factor, GC, is defined as the weighted sum of capability indices for all desired features. The index values are 0 if a desired feature was not available, 1 if a desired feature was available, and -1 if a required or critical feature was not available. The weights indicated the relative importance of each feature. 7h

The degree of interaction is measured by the interaction quotient, IQ, and is defined as a function of the ratio of the effective data rate of the connecting channel to the interaction data flow rate requirement. The IQ is expressed as a weighted sum of quotient terms for each type of interaction which could occur. Each term has the form DR/(DO+DI), where DR is the effective data rate of the channel and DO and DI are the output and input data rates, respectively, at the graphics console required to request and receive response for that type of interactive service request. 7i

Direct costs are the pro-rata portions of system resources attributable to graphics support. These include the fair share of all remote, interconnecting and local hardware and software that is necessary to implement the graphics system. 7j

Finally, the amplification factor, A , was defined as follows:

$$A = \frac{d1\ P\ (d2\ IQ + d3\ GC)}{d4\ \$/C}$$

where P is the productivity factor; IQ is the interaction quotient; GC is the graphics capability; $ is the total system direct dollar costs; and C is the number of graphic consoles that may be concurrently active in the system. The di are the assigned weights that reflect the significance attached by installations towards each of these factors. A is thus a weighted measure of the change in the rate of productive output per dollar of total graphics system direct cost per active console. 7k

Dunn's approach may be reduced to a form closer to the simpler ratio desired for cost-benefit analysis. Performance is actually a function of both the interaction quotient and graphic capability. Thus, the figure of merit reduces to a performance function over a cost function. All that is lacking is to relate performance to value, but this is facilitated by the way in which performance is expressed. By expressing performance in terms of productivity gains, it is easy to determine value by considering the value of performance achieved the old way. The benefits of the new system are just the unit value of output done the old way times the rate of increase of productivity. 7l

The real criticism of Dunn's approach is that it is circular: it starts with the data it should be seeking. The problem is to determine the productivity gains from using a particular system. If this is known, and the cost for using the system can be determined, then the cost-benefit ratio for the system can be determined. 7m

Conclusions 8

The quantitative analysis of computer graphics systems is performed at a very crude level of sophistication today. This is partially because it has a basis in the evaluation of computer systems in general, which is still a very inexact science, and partially because it deals with a system involving humans, whose performance is always difficult to assess. 8a

As we have explained, performance evaluation of computer systems in general and graphics systems in particular can be done reasonably well. Cost analysis can be done well also; consequently so can cost-effectiveness analysis. On the other hand, benefit evaluation cannot yet be done very well, so neither can cost-benefit analysis. 8b

An experimental approach was outlined which seeks to apply the same types of tools to benefit analysis which have been applied to performance analysis. Benefit analysis is the area which must be improved before cost-benefit analysis can be employed in any meaningful fashion. 8c

Even then, cost-effectiveness analysis will continue to be the principal tool for choosing among alternative systems on the same order of performance. Cost-benefit analysis is too gross a tool to detect small differences between systems. Cost-benefit analysis will be increasingly used, however, to place the "go or no-go" investment decision on a more analytic basis. 8d

BIBLIOGRAPHY

9

AU68 Auerbach Info, Inc. "Design and applications of automated display systems." <Auerbach Standard EDP Reports>, Special Report, September 1968 9a

BO71 Boehm, B. W., Seven, M. J. and R. A. Watson. "Interactive problem-solving -- An experimental study of 'lockout' effects." <Proc.> Spring Joint Computer Conference, 1971, pp. 205-210. 9b

CA67 Calingaert, Peter. "System performance evaluation: survey and appraisal." <Communications of the ACM>, 10:1 (January 1967), pp.12-17. 9c

CA70 Carlisle, James H. "Comparing behavior at various computer display consoles in time-shared legal information." RAND Corporation Report P-4448, September 1970. 9d

CI72 Cislo, Robert A. "Graphic system performance evaluation." <Proc.> 1972 ACM National Conference, pp. 432-442. 9e

CO72 Cotton, Ira W. "Network graphic attention handling." <Proc.> Online 72 Symposium, Brunel University, September 1972. 9f

DU73 Dunn, Robert M. "Computer graphics: capabilities and usefulness." <Proc.> SHARE XL (Denver, Colorado, 7 March 1973). 9g

FO69 Foley, James D. <Optimum design of computer driven display systems>. Doctoral Dissertation, University of Michigan, March 1969. 9h

FO71 Foley, James T. "An approach to the optimum design of computer graphics systems." <Communications of the ACM>, 14:6 (June 1971), pp. 380-390. 9i

GO69 Gold, Michael M. "Timesharing and batch-processing: an experimental comparison of their values in a problem-solving situation." <Communications of the ACM>, 12:5 (May 1969), pp. 249-259. 9j

HI69 Hillegass, John R. "Systematic techniques for computerevaluation and selection." <Management Services>, July-August 1969, pp. 35-38. 9k

JA71 Jacobs, Lesley D. "CRT graphics consoles - an aid to

selection." Rome Air Development Center Technical Report 71-61, November 1971. 9l

JOH70 Johnson, R. R. "Needed: a measure for measure." <Datamation>, December 15, 1970; pp. 22-30. 9m

JO71 Joslin, Edward O. "Costing the system design alternatives." <Data Management>, April 1971, pp. 23-27. 9n

JO64 Joslin, Edward O. and Martin J. Mullin. "Cost-value technique for evaluation of computer system proposals." <Proc.> Spring Joint Computer Conference, 1964, pp. 367-381. 9o

LA66 LaBolle, V. "Development of equations for estimating the costs of computer program production." System Development Corporation, June 1966 (Published as U. S. Air Force Electronic System Division Technical Report No. 66-350). 9p

MI68 Miller, Robert B. "Response time in man-computer conversational transactions." <Proc.> Fall Joint Computer Conference, 1968, pp. 267-277. 9q

MY68 Myer, T. H. and I. E. Sutherland. "On the design of display processors." <Communications of the ACM>, 11:6 (June 1968), pp. 410-414. 9r

NE67 Nelson, E. A. "Management handbook for the estimation of computer programming costs." System Development Corporation, Technical Memorandum No. 3225, March 20, 1967. 9s

PO69 Poole, Harry H. "Computer display system tradeoffs." In <Computer Graphics: Techniques and Applications>, Eds.: Parslow, Prowse & Green. New York: Plenum Press, 1969. 9t

PR71 Prince, M. David. <Interactive Graphics for Computer-Aided Design>. Reading, Massachusetts: Addison-Wesley, 1971. 9u

SH69 Sharpe, William F. <The economics of computers>. Columbia University Press: New York, 1969. (RAND Report R-463-PR). 9v

SM68 Smith, J. Meredith. "A review and comparison of certain methods of computer performance evaluation." <Computer Bulletin>, 12:1 (May 1868), pp. 13-18. 9w

TH67 Thomas, E. M. "System considerations for graphic data processing." <Computers and Automation>, November, 1967, pp. 16-19. 9x

WA47 Ward, John E. "Systems engineering problems in

computer-driven CRT displays for man-machine communication."
<IEEE Trans. System Science and Cybernetics>, 3:1 (June 1967), pp.
47-54. 9y

9z

TABLE OF CONTENTS 10

Cotton: Cost-Benefit Analysis of Computer Graphics Systems

(J18387) 13-AUG-73 13:21; Title: Author(s): Ira W. Cotton/IWC;
Distribution: /KLB; Sub-Collections: NIC; Clerk: IWC;
Origin: <NBS-TIP>GRAPHICS-COTTON.NLS;2, 13-AUG-73 11:58 IWC ;

Please do a "type <mit-multics>lims.runout" from exec level to see
the proposed addition to the NETED spec. 1

(J18388) 13-AUG-73 13:27; Title: Author(s): Michael A. Padlipsky/MAP; Distribution: /USING; Sub-Collections: NIC USING; Clerk: MAP;

/asdesign
Virtual Machines, the Kernel, and the Virtual Machine Monitor 1

As discussed earlier, it is our goal
to build a proven secure virtual machine operating
system for the PDP11/45. It is worthwhile here
to review the differences between a normal operating system and a
virtual machine system. 2

A normal operating system runs on the bare machine and produces
environments in which user jobs run. These environments
usually consist of some memory, and the ability to execute
most of the instructions of the bare machine. However, the
priviledged instructions of the bare machine have
usually been removed and a set of new extended instructions
(supervisor calls) have been added to provide
user services (i/o, etc.) A virtual machine system
also provides a user job an environment but here
the environment looks logically like the original bare
machine environment. This environment is typically
implemented by using the real machines relocation (or mapping)
hardware to provide a memory which addresses from 0
like the bare machine, and to run
users in non-priviledged mode, trapping priviledged instructions and
simulating their effects. Thus, the environment looks to the user
job like the bare machine except that it is
slower (timing do to instruction simulation of priviledged
instructions). (see popek and goldberg) (see figure) 3

In order to build a secure system , we will separate that part of the
system required for security in to a Kernel, which will be proven
correct. The kernel must arbitrate all accesses
to resources. (see popek paper)
The rest of the system will be called the Virtual Machine Monitor
(VMM). The users jobs will run on virtual machines and will be
called virtual machines.
(see figure) 4

The PDP11/45 5

The PDP11/45 is not suitable as presently stands for
virtualization (see popek and goldberg). However, our
PDP11/45 will be modified to make it suitable.
The major changes are that certain sensitive instructions
have been made priviledged. 6

On the PDP11/45 there are three modes: kernel, supervisor,
and user. Each mode is protected from the others.
Thus, it is natural to run the Kernel in kernel mode, the

VMM is supervisor mode, and the virtual machines in
user mode. The Kernel mode will be the only mode
allowed to perform certain instructions (sensitive) so
that the kernel will be able to enforce the security
decisions. 7

The Virtual Machine Monitor (VMM) 8

The VMM is a large piece of code which we will not be able to prove
correct. However, this fact is not a security issue
in itself. If the VMM makes errors, the worst it should be able to
do is damage the operation of some user's virtual machine.
Thus in the general case, we are not interested in the design of the
VMM. 9

However, we are interested in guaranteeing the security of the
system. Especially, a problem called the Morse Code Problem
(see ...). The idea is that if two
users who are not supposed to be able to communicate
can send one bit to each other through some complicated
mechanism, then, they can send an arbitrary message
by the use of morse code (or other code say EBCDIC but
Morse code sounds better).
It turns out that the design of the VMM does effect this issue. 10

Suppose the VMM has a bug in it such that user A can do something in
his virtual machine which causes the VMM to store a bit. Later when
user B runs, he may be able to cause the VMM to give him that bit.
Thus, the two users can send messages from
one to the other. 11

Our solution to this problem was so obvious to us
after we thought of it, that we were supprised how long it took us.
The solution is to divide the VMM into pieces. In this case, the VMM
was divided into a simulator (which performs the
virtual machine priviledged instruction simulation), a CPU
scheduler (which handles cpu scheduling and memory
management), a disk scheduler (which handles
disk requests because the disk is shared), etc.
Each virtual machine also has with it an associated simulator. These
simulators each have there own local writeable storage
but there is no sharing of this local storage between two
simulators.
The code might be shared but it must be execute only.
Thus, the two simulators for user A and user B have no common
storage. Even if user A can get a bit to his simulator, user B's
simulator can not get this bit because there is no shared storage,
and thus, user B can not get this bit.
Thus, the simulators can only communicate throught the Kernel and the

Kernel can check the communication which is
exactly what we want. 12

For the same reason, the disc scheduler (or any shared I/O device
scheduler), and the CPU scheduler are separate processes with their
own local storage. 13

The Simulator 14

The Simulator performs most of the functions necessary to
create a virtual machine environment. The simulator
runs in supervisor mode on the PDP11/45 while
the user's virtual machine runs in user mode. We
have had the PDP11/45 modified so that traps while in
user mode go to the supervisor directly, thus to
the simulator directly with no Kernel intervention.
Simulator traps go to the Kernel. Thus when the
virtual machine executes an I/O instruction, the
Simulator gets control. It can then decode the I/O, perform
what ever it wants to do to simulate the I/O, and if
the I/O should cause real I/O to occur, then
the simulator calls the Kernel to perform the real
I/O. The Kernel can then check to be sure
the user is allowed to do I/O to this device, and that
what he is doing is allowed. 15

The simulator will need to be able to do certain things to
the virtual machine, like fetch/store into its memory,
change processor state, etc. These operations will be done by calls
to the Kernel which can check there validity and perform them. 16

The kernel often needs to send replies to the simualator at
asynchronous times. For example, when an I/O completes
and an interrupt comes in, the kernel wants to notify the
simulator even if it is not in core.
One approach would be for the kernel to bring the
simulator into core but this means more
code to prove and its implies a scheduling
discipline. The approach we have chosen is that there is
a locked down (incore) queue for each simulator
(or other process) into which the kernel can dump messages
for the process when it has them. Thus, the kernel does not
need to have a queue of its own to save stuff for later. These
queues are sharable between a process and the kernel.
The kernel must guarantee that a process can only access
its own queue (part of invoking a process). 17

The CPU Scheduler 18

The CPU scheduler performs scheduling functions and memory management functions. It gets data from the Kernel about which jobs want to run. It makes a decision, but for security, must request that the Kernel actually switch the CPU to the proper job. Similarly, when the CPU scheduler decides to swap a segment in or out of memory, it must request the Kernel to perform this so that it can not swap something out for one user, and read it back for another, thus allowing illegitimate passage of information. Notice that the cpu scheduler never passes information back to a process. Thus, it is an information sink.
This property means that we can allow it to have read access to lots of information (for example, how much cpu time each task has had) so that the scheduler can be fairly sophisticated about its scheduling disciplines. We suspect for a first scheduler, the scheduler will give priority to ANTS and run other virtual machines sort of in the background.
Also notice that the scheduler performs all its operations (scheduling and swapping) through the Kernel. Like the simulators, it does not need to be proven correct, for the worst that will happen is some job won't run. 19

The disc scheduler 20

The disc is a shared device. However, we will make it appear as a lot of separate mini-discs to the VMs. Each VM thinking it has a very small disc. When two VMs try to use the disc at the same time, we must schedule them. It has been shown (see ...) that if the same device scheduler is used for two devices, it can allow morse code in the ordering of requests. However, in our case, we will have separate device schedulers (with no shared writeable storage) for each shared device. Also, each VM may have at most one request pending. Thus, no ordering of requests by the disc scheduler passes information back to the VMs. (Of course, if a VM has access to a real time clock, he may be able to communicate to another VM if the disc scheduler and CPU scheduler can be coerced into ordering requests. WE will ignore this problem and one could prove the code of the schedulers later independently to show that this can not happen).
The actual request by the VM is kept with the VM. The disk scheduler may look at it but not change it. Thus it is also an information sink, and may thus be built with sophisticated scheduling algorithms (say based on seek address). 21

Shared Pseudo Devices 22

We will allow some users to communicate through shared pseudo devices. This means that when one VM attemts to use

some I/O device, I/O is not really
performed but instead a message is passed to another VM.
Two users may set up a shared pseudo device only if the
Kernel allows them to and this will be based on the security
data. 23

We could build the mechanism for shared pseudo devices
into the Kernel. However, this would probably limit
the choices of pseudo devices to a very simple one which could be
easily proved. Another mechanism, would be to allow
two simulators to share a segment, and then to simulate
to the VM a pseudo device.
This mechanism has the advantage that we don't have
to prove its correctness, and the simulators can
provide any pseudo device which is convenient
for the VM. For example, ANTS normally has teletypes which
allow people to access the network. To allow another VM
access to the network would require no changes
to ANTS for we could simply tell its simulator to simulate a
pseudo device which was a virtual teletype.
Then the other virtual machine which wants to use the network talks
to its simulator through a pseudo device (not necessarily a
teletype), the two simulators talk through a shared segment, and the
simulator communcates to ANTS through a pseudo teletype. 24

There are some theoretical problems with the shared
segment approach.
If the Kernel performed shared pseudo devices,
then the user could
check his input from the pseudo device for validity
before taking some action. But if the pseudo device is handled by
the simulator, it is possible that a bug in the simulator
will cause one user to be able to destroy another
(write in his memory say). We do not believe that this
will be the case since
the simulator will be well written, but since we will not
prove the simulator, it is conceptually possible.
However, we have come up with a design where the user
writes the pseudo device handler (or maybe we write
a proved few) which run in separate protected
process from the simulator. Then it can have the
shared segment and can check for validity before passing
the data to the simulator. We will explore this
design later but for now, we will use the simpler case.
(Note: users who do not use pseudo devices are
still protected and I emphasize it takes a bug in the
simulator for damage to occur). 25

The Front End 26

The front end is a process which handles the identification problem. This process initially holds all teletypes. When a user walks up to a teletype, he signs in, identifies himself, and requests to be connected to an existing virtual machine (say ANTS) or to create a new virtual machine (say DOS). The front end then requests the Kernel to perform the requested operation for this user. Conceptually, the front end must be proved. It must be shown that it correctly identifies the user and always makes it requests for the proper user so that the Kernel can perform the check. We will not attempt to solve the identification issue here (it is very hard), but for now will use something like name and password. We will prove that the front end after getting a name will correctly use it in all dealings with the Kernel. (We will have already proved that once the Kernel is asked to do something by the front end, it will only perform it if the security data allows the operation). 27

The Simulator will be built with its own escape convention. That is, what you type to tell the simulator that your VM has gone crazy and you want the simulator to do something will be up to the designer of the simulator. However, if your simulator breaks, you will have to go to a different terminal, connect to the front end to get your VM/Simulator killed. This mechanism allows us not to have to build a proven escape mechanism in the kernel. We could, of course, later prove the simulator (we won't) and then you would never need to go to another teletype. This is just a convenience issue and not a security issue and that is why the choice was made. 28

The Updator 29

The mechanism by which one updates the security data to, for example, add a new user, is to talk to the Kernel and request it to perform updates to its security data (with whatever checks we build in). However, if one could do this from his virtual machine/simulator pair, one could never guarantee the integrity of the security data. For example, suppose you said "add JOE with ability to use ANTS but not DOS". The simulator (through a bug or whatever) might send to the Kernel "add JOE with ability to use ANTS and DOS" or even "add BILL" and you might not be able to detect this. Even questions like tell me about JOE might respond with the information about Bill because of a simulator bug.
The two solutions to this problem are either prove the correctness of

the simulator or build a correct updator. We will
choose the later since we believe it is doable. That is,
we will build a proven correct updator which is used
to update the security data. As a first cut, this updator
will be a particular teletype which is in
a locked room or something (maybe has a guard, at least
conceptually it could) which has been
granted access to update the security data. Some kernel
checks will still be made on requested updates to make
sure that the kernel is still secure (see ...) 30

Thus the strucure of the Kernel, VMM, VM is shown in
figures ...
Parts with double lines arround them are proven correct. 31

32

(J18390) 13-AUG-73 14:28; Title: Author(s): Chuck S. Kline/CSK ;
Distribution: /; Sub-Collections: NIC; Clerk: CSK;

JCP 13-AUG-73 16:30 18392

recycle printer paper

we are once again going to try to recycle printer paper. put white printer or teletype paper, blank or printed on, in the boxes outside of and accross from the printer room door. please try to stack the paperin carefully so that it doesn't take up so much room. we can't use the green cans anymore so we will have to make do with empty boxes. thanx.

1

recycle printer paper

(J18392) 13-AUG-73 16:30; Title: Author(s): Jeffrey C. Peters/JCP;
Distribution: /SRI-ARC; Sub-Collections: SRI-ARC; Clerk: JCP;

JCN 13-AUG-73 19:09 18393

SOW Changes by JCN

Duane: This is a parallel message. The sow is edited by jcn and in (norton,arpautil,). I sent a sndmsg, too. See (norton,arpautil,jcn) for the changes I made. Thanks, Jim 1

SOW Changes by JCN

(J18393) 13-AUG-73 19:09; Title: Author(s): James C. Norton/JCN;
Distribution: /DLS; Sub-Collections: SRI-ARC; Clerk: JCN;

ANTS Command Language

Dear Dave; 1

Sorry, to be so long reponding to your note; but things have been pretty hectic around here the last few weeks. Now about your suggestions.....First, unfortunately your assumption that the new ANTS will resemle the old is misplaced. I dont know if you realize it but MARK I is a KLUDGE. It was forced on us by higher up in order to get on the net, and then RJE to CCN , etc. So now we are doing what we really want. 2

We are a line at a time system. Our approach to command implementation is very close to the method used by Multics. I admit there are some very nice features to the TENEX approach, but it violates several of our desgn criteria. 3

With reference to BACK, DELETE, and other similar operations. Yes, of course we will do it the way you suggest. (Come now we arent that dumb.) 4

The note about the flag character is not applicable to MARK II ANTS. Getting commands to ANTS is handled entirely different. 5

Similarly, we are planning to have several sorts of status commands to get information about users, peripherals, etc. I dont expect the initial version will go so far to attempt a connection to a site when you ask for host status, but later we probably will. 6

Sorry, but I imagine the TO command will remain both to device and to usercode. (Presently, there is no such thing as logon to MARK I ANTS, which is why names dont appear on a status table. 7

Teach will become HELP. 8

We will allow local echo as you describe. 9

You will be able to flush output to any device you have control over and be able to re-route information to other devices. 10

About the TIP emulator, we might do it if we find that there are enough masochists who want it but I expect it may be difficult to find someone willing to do it. 11

I am convinced that no matter what names we came up with for commands we couldn't satisfy everyone (although I admit UP is pretty bad), therefore the command interpreter allows as many synonyms or aliases for a command as you want. I imagine the system will come with a basic set of names of commands which will form a common basis for all ANTS systems and then you can add all the aliases you want to fit your whim. Also we have tried to make it as easy as possible to

write your own commands. (So if you guys start giving us to bad a time we will tell you to go write your own.)

About all those commands that arent defined. They are mostly related to the various kludges mentioned before. They wont have any bearing on MARKII.

The first versions of ANTS will not have all the bells and whistles we want. But most of them are in being planned. Once we can use the disk to store files, we want to make the command interpreter work very much like Multics.(ie. type a name and run thru the disk directory to find a code file with the same name and execute it.) And all sorts of other goodys. However, there are several other things that must take precedence and we will get to them as soon as we can. Please feel free to drop me a line when ever you have a question or a suggestion. I wont guarantee to know the answer or agree with you, but I will listen and consider it or try to find out. Why just think: On this batch I agreed with you on over half of the suggestions, which is pretty good.

Take care and keep in touch,
John

ANTS Command Language

(J18394) 13-AUG-73 19:57; Title: Author(s): John D. Day/DAY;
Distribution: /DHC; Sub-Collections: NIC; Clerk: DAY;
Origin: <ILLINOIS>ANTSCMD.NLS;1, 13-AUG-73 18:44 DAY ;

DVN 14-AUG-73 08:51 18395

Please Try Printing Again

I am sending this via journal because I could not get through to you via phone

DVN 14-AUG-73 08:51 18395

Please Try Printing Again

Elizabeth Micheal believes she has fixed the bug that was botherng printing through your terminals, but she would like to check it under actual use. Could you try printing something as you did bdfore except specify O D E (for output device experimental) instead of ODT, ad let us know what happens?
thanks
.

1

18395 Distribution
Elizabeth K. Michael, John S. Perry,

Please Try Printing Again

(J18395) 14-AUG-73 08:51; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /EKM JSP; Sub-Collections: SRI-ARC; Clerk: DVN;

VGC 14-AUG-73 11:07 18396

Changes to INWG mailing list

Marcia: Here are a few changes and additions for INWG. 1

New INWG members who should receive back copies of relevant documentation 2

(INWG Notes 2,4,5,6,10,11,13-26,29,31-33) 2a

(or if you prefer NIC numbers: 2b

12396,12520,12535,13634,13636,13653,13654,13865,13879,14188,14133,13774,14203,14497,14498,14741,15184,16075,16076,16077,16429,16735,17353,17364) 2b1

Add the following new names (to INWG list only): 2c

Paul Baran 2c1

Cabledata Associates, Inc. 2c1a

701 Welch Road, Suite 326 2c1b

Palo Alto, Calif 94304 2c1c

(415) 328-2411 2c1d

Dr.-Ing. Georg Faerber [spelling is correct) 2c2

D-8012 Ottobrun 2c2a

Egerweg 3 2c2b

West Germany 2c2c

Note: Faerber is self-employed consultant 2c2c1

Hartmut Grebe 2c3

815 Indian Rock Ave. 2c3a

Berkeley, Calif 94707 2c3b

H. J. Schneider 2c4

Institut fuer Informatik 2c4a

7 Stuttgart 2c4b

Herdweg 51 2c4c

Changes to INWG mailing list

West Germany 2c4d

Dr. Eric Foxley 2c5

Director of Computing, Cripps Computing Center 2c5a

University of Nottingham, University Park 2c5b

Nottingham NG7 2RD 2c5c

United Kingdom 2c5d

Harold C.Folts 2c6

National Communications System 2c6a

8th Street and S. Courthouse Road 2c6b

Arlington, Virginia 22204 2c6c

Christopher Newport 2c7

Telenet Communications, Inc. 2c7a

1666 K. Street NW 2c7b

Washington, D.C. 20006 2c7c

(202) 785-8444 2c7d

D. J. Blackwell 2c8

Manager,Corporate Relations and Standards 2c8a

International Computers Limited 2c8b

ICL House, Putney 2c8c

London SW15 1SW 2c8d

United Kingdom 2c8e

01-788-7272 2c8f

T. Nakajo 2c9

Manager, Electronic Switching System Dept. 2c9a

Fujitsu, Ltd. 2c9b

Changes to INWG mailing list

1015 Kamikodanaka 2c9c

Nakahara-ku, Kawasaki 2c9d

Japan 2c9e

(044) 77-1111 2c9f

Takashi Uetake 2c10

Japan Telephone and Telegraph 2c10a

680 5th Avenue 2c10b

New York, N.Y. 10019 2c10c

(212) 586-7634 2c10d

Prof. David Farber 2c11

Computer Science Dept. 2c11a

University of California 2c11b

Irvine, California 92664 2c11c

(714) 833-6891 or 5233 2c11d

Prof. Richard Grimsdale 2c12

University of Sussex 2c12a

School of Applied Sciences 2c12b

Brighton BN1 9QT 2c12c

United Kingdom 2c12d

02-736-6755 2c12e

Ira Cotton 2c13

National Bureau of Standards 2c13a

Building 225, Room B216 2c13b

Washington, D.C. 20234 2c13c

H. C. Way 2c14

Group Telecommunications Planning Manager 2c14a

British Airways Board 2c14b

P.O. Box 13 2c14c

Air Terminal 2c14d

Buckingham Palace Road (Elizabeth Bridge Entrance) 2c14e

London SW1W 9SR 2c14f

United Kingdom 2c14g

01-828-6822 2c14h

A. M. M. Thomson 2c15

Technical Services Division 2c15a

Central Computer Agency 2c15b

River Walk House 2c15c

157/161 Milbank 2c15d

London SW1P 4RT 2c15e

United Kingdom 2c15f

01-828-8040 X397 2c15g

Sr. Ysmar Vianna e Silva Filho 2c16

Head, Computing Center 2c16a

Nucleo de Computacao Eletronica 2c16b

Universidade Federal do Rio de Janiero 2c16c

Caixa Postal 2324-ZC-00 2c16d

20000 - Rio de Janiero - GB 2c16e

Brasil 2c16f

Hidetoshi Kawai 2c17

Computer Division 2c17a

Changes to INWG mailing list

Electrotechnical Laboratory 2c17b

Nagata-cho, Chiyoda-ku 2c17c

Tokyo, Japan 2c17d

581-0441 (Tokyo) 2c17e

Changes to be made to existing mailing list entries: 3

Barry D. Wessler to new address 3a

Telenet Communications, Inc. 3a1

1666 K St. NW 3a2

Washington, D.C. 20006 3a3

(202) 785-8444 3a4

Finally, please add my name to the Packet Radio mailing list and arrange to me to get back copies of the Packet Radio Notes. 4

Thanks very much, Marcia. Holler if you have questions. Please do the best you can to get this update done soon. I want to send out meeting notices for a September 16 session in London and would like to use the new mailing list. Cheers. Vint Cerf 5

18396 Distribution
Marcia Lynn Keeney,

Changes to INWG mailing list

(J18396) 14-AUG-73 11:07; Title: Author(s): Vinton G. Cerf/VGC;
Distribution: /MLK; Sub-Collections: NIC; Clerk: VGC;
Origin: <SU-AI>INWGUPDATES.NLS;5, 14-AUG-73 10:56 VGC ;

KTP 14-AUG-73 11:22 18397

Jim -- last week I tried to send you a lengthy (several pages) piece of Journal mail from Multics. Have you gotten it? I received no notification of an Author Copy, so I am skeptical. The mail was about my experiences with (and suggestions for) Network Journal submission and delivery. I'd appreciate it you could let me know if you got it. -- Ken 1

18397 Distribution
James E. (Jim) White,

(J18397) 14-AUG-73 11:22; Title: Author(s): Kenneth T. Pogran/KTP ; Distribution: /JEW ; Sub-Collections: NIC; Clerk: KTP;

DHC 14-AUG-73 11:50 18398

What's in a name, besides clarity?

Mike -- I consider it very important that the proposed editor be called NETEDS. This may seem picayune, but it was the basis for my retracting my other objections, some time back. If we are going to offer this as an interim, simplistic, or whatever solution to a need, let's not give it such an officious title.

--dave. 1

18398 Distribution
Michael A. Padlipsky, Nancy J. Neigus,

DHC 14-AUG-73 11:50 18398

What's in a name, besides clarity?

(J18398) 14-AUG-73 11:50; Title: Author(s): David H. Crocker/DHC;
Distribution: /MAP NJN; Sub-Collections: NIC; Clerk: DHC;

Jim -- 1) What happens when Group limits are changed and a group already has too many users (for the new limit)?

2) Forgot the second question.

--dave

1

18399 Distribution
James C. Norton,

(J18399) 14-AUG-73 11:53; Title: Author(s): David H. Crocker/DHC;
Distribution: /JCN; Sub-Collections: NIC; Clerk: DHC;

Dean -- Thanks for the notes and info on SYSGD. (Am just responding because i just got back from vacation.) It would be very useful if SYSGD had more comments and some sort of index to the functions. The actual function names are a bit criptic. With the addtion of such aids, SYSGD would be incredibly useful.

(Dont know yet when I'll be able t.)

tnx --dave.

1

18400 Distribution
N. Dean Meyer,

(J18400) 14-AUG-73 11:57; Title: Author(s): David H. Crocker/DHC;
Distribution: /NDM; Sub-Collections: NIC; Clerk: DHC;

Another Look at Superwatch vs. the Fact Files

During the last four weeks, the average difference between the CPU used according to Superwatch and the Fact Files is 18%. 1

This compares to 11.5% for the preceding four weeks. 2

If you think this warrants taking some action, let me know. 3

18401 Distribution
Paul Rech,

SRL 14-AUG-73 12:05 18401

Another Look at Superwatch vs. the Fact Files

(J18401) 14-AUG-73 12:05; Title: Author(s): Susan R. Lee/SRL;
Distribution: /PR; Sub-Collections: SRI-ARC; Clerk: SRL;
Origin: <LEE>BLAP.NLS;1, 14-AUG-73 11:59 SRL ;

Dean -- Dont know if message i just sent you got sent, since my Net connection died. Anyhow, thanks for notes and info on L10 (just vgot back from vacation). It would be very useful to have SYSGD have more comments and also be indexed. The function names are a bit too criptic for comprehension.

tnx. --dave.

1

18402 Distribution
N. Dean Meyer,

(J18402) 14-AUG-73 12:06; Title: Author(s): David H. Crocker/DHC;
Distribution: /NDM; Sub-Collections: NIC; Clerk: DHC;

Nancy Niegus should be sending us the info on re-enableing TIP terminals. By the way, I don7t use Binary Output and haven't noticed any problems.

Anyhow, suggestion: Have the imlac put the circles on the sreeen, whenever CA hit on bug. That way the user get very quick feedback on his pointing and can immediately correct.. NIC would still do the deletes. Thoughts?

--dave.

1

18403 Distribution
Kenneth E. (Ken) Victor,

(J18403) 14-AUG-73 12:10; Title: Author(s): David H. Crocker/DHC;
Distribution: /KEV; Sub-Collections: NIC; Clerk: DHC;

Utility Personnel You Should Know

You will be seeing two new faces around ARC, especially in the next few months. These people are Mike Marrah and Jim Blum. Mike will be managing the utility system over at Tymshare, and Jim will be the system programmer for it. If any of you have any questions or suggestions on the utility, or are simply interested in some aspect of it, please feel free to talk with them. To send info to them via SNDMSG or the Journal, the appropriate destinations are <MARRAH> or MLM, and <JIMB> or JIMB.

1

18404 Distribution

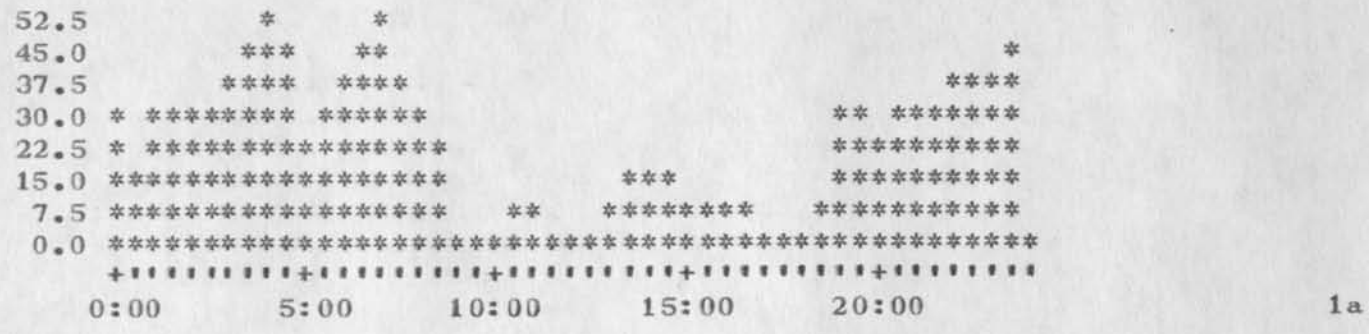
Michael L. Marrah, A. Jim Blum,
Jeanne M. Leavitt, Rodney A. Bondurant, Jeanne M. Beck, Mark Alexander Beach, Judy D. Cooke, Marcia Lynn Keeney, Carol B. Guilbault, Susan R. Lee, Elizabeth K. Michael, Charles F. Dornbush, Elizabeth J. (Jake) Feinler, Kirk E. Kelley, N. Dean Meyer, James E. (Jim) White, Diane S. Kaye, Paul Rech, Michael D. Kudlick, Ferg R. Ferguson, Douglas C. Engelbart, Beauregard A. Hardeman, Martin E. Hardy, J. D. Hopper, Charles H. Irby, Mil E. Jernigan, Harvey G. Lehtman, Jeanne B. North, James C. Norton, Jeffrey C. Peters, Jake Ratliff, Edwin K. Van De Riet, Dirk H. Van Nouhuys, Kenneth E. (Ken) Victor, Donald C. (Smokey) Wallace, Richard W. Watson, Don I. Andrews

Utility Personnel You Should Know

(J18404) 14-AUG-73 13:45; Title: Author(s): Ferg R. Ferguson/WRF;
Distribution: /SRI-ARC MLM JIMB; Sub-Collections: SRI-ARC; Clerk: WRF;

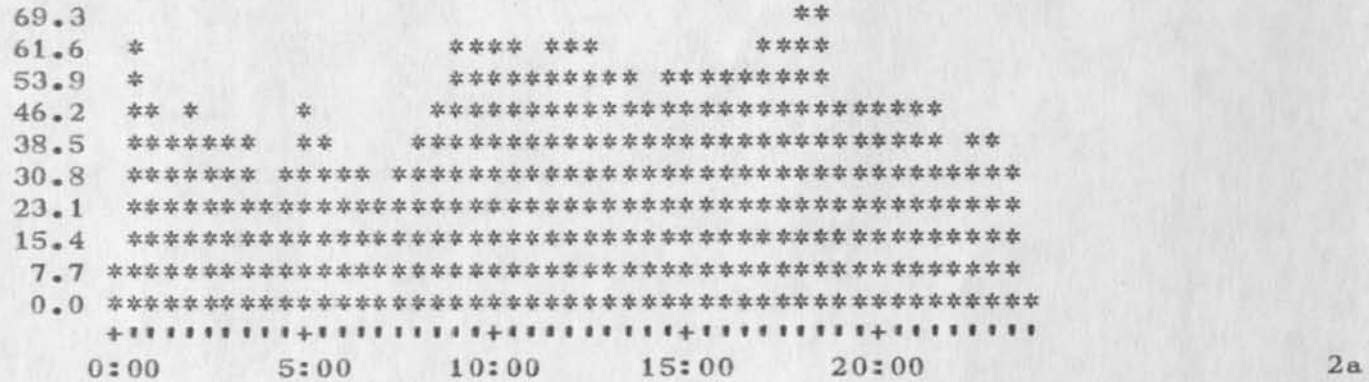
Superwatch Average Graphs for Week of 8/6/73

TIME PLOT OF AVERAGE IDLE TIME FOR WEEK OF 8/6/73
x axis labeled in units of hr:min, xunit = 30 minutes 1



1a

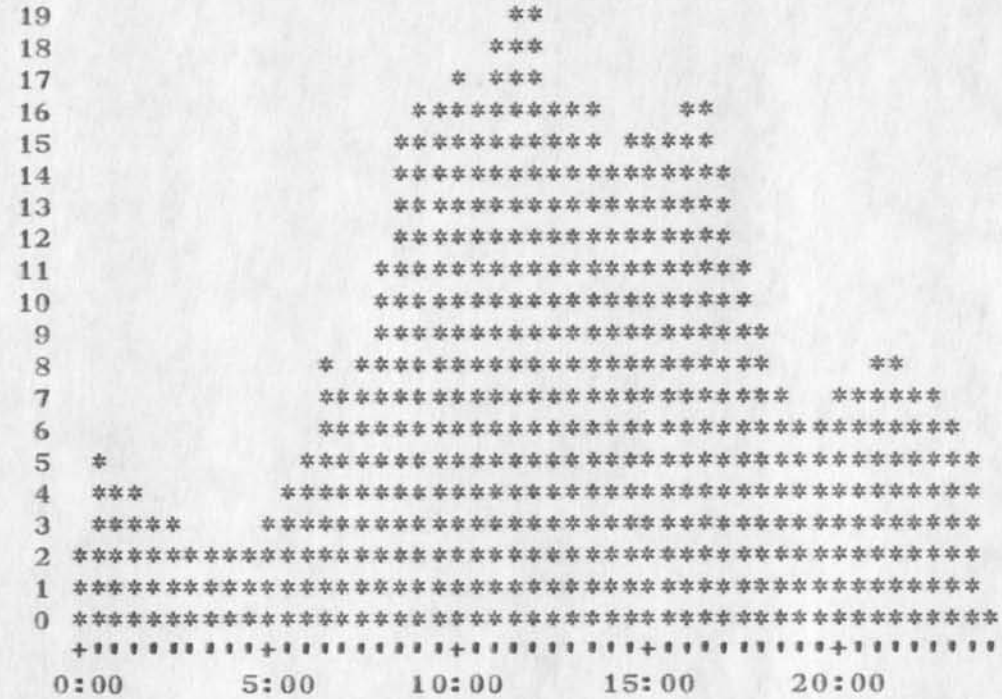
TIME PLOT OF AVERAGE PER CENT OF CPU TIME CHARGED TO USER ACCOUNTS
FOR WEEK OF 8/6/73
x axis labeled in units of hr:min, xunit = 30 minutes 2



2a

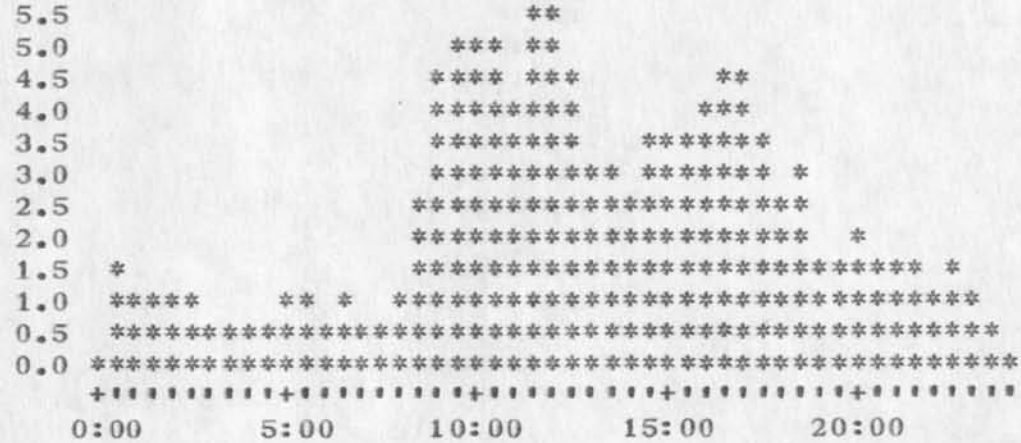
Superwatch Average Graphs for Week of 8/6/73

TIME PLOT OF AVERAGE NUMBER OF USERS FOR WEEK OF 8/6/73
x axis labeled in units of hr:min, xunit = 30 minutes 3



3a

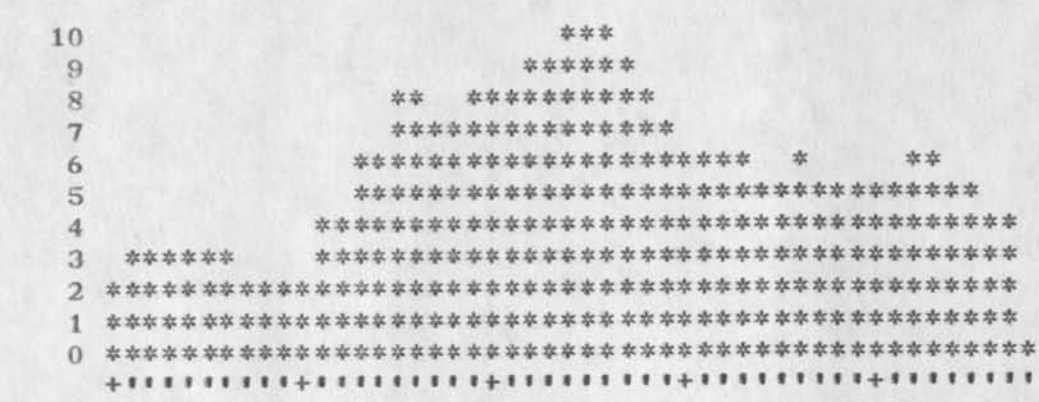
TIME PLOT OF AVERAGE NUMBER OF GO JOBS FOR WEEK OF 8/6/73
x axis labeled in units of hr:min, xunit = 30 minutes 4



4a

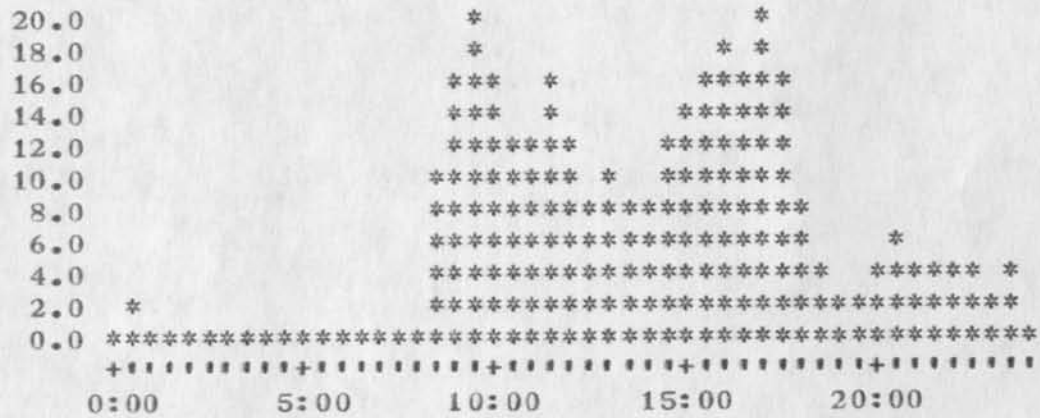
Superwatch Average Graphs for Week of 8/6/73

TIME PLOT OF AVERAGE NUMBER OF NETWORK USERS FOR WEEK OF 8/6/73
x axis labeled in units of hr:min, xunit = 30 minutes 5



5a

TIME PLOT OF AVERAGE PER CENT OF SYSTEM USED IN DNLS FOR WEEK OF 8/6/73
x axis labeled in units of hr:min, xunit = 30 minutes 6



6a

Superwatch Average Graphs for Week of 8/6/73

(J18405) 14-AUG-73 14:53; Title: Author(s): Susan R. Lee/SRL;
Distribution: /JCN RWW DCE PR DCW JCP DVN JAKE CFD KIRK DLS BAH;
Sub-Collections: SRI-ARC; Clerk: SRL;
Origin: <LEE>WEEK8/6GRAPHS.NLS;2, 14-AUG-73 14:50 SRL ;

18405 Distribution

James C. Norton, Richard W. Watson, Douglas C. Engelbart, Paul Rech, Donald C. (Smokey) Wallace, Jeffrey C. Peters, Dirk H. Van Nouhuys, Elizabeth J. (Jake) Feinler, Charles F. Dornbush, Kirk E. Kelley, Duane L. Stone, Beauregard A. Hardeman,

NJN 14-AUG-73 14:55 18406

Re NETED

Dave--
I don't understand what the 'S is for in NETEDS. Personally I prefer NETED and changed the USING Meeting notes to say so. (Too late, they are already being published that way.) And waht is officious about NETED anyway? I think you need some better reasons than you have explained. --Nancy 1

18406 Distribution
David H. Crocker, Michael A. Padlipsky,

Re NETED

(J18406) 14-AUG-73 14:55; Title: Author(s): Nancy J. Neigus/NJN;
Distribution: /DHC MAP; Sub-Collections: NIC; Clerk: NJN;

From: Padlipsky.CompNet at MIT-Multics 08/14/73 1754.9 edt Tue

I concur.

Will spring that as separate ("trivial") issue when we go to press. I.e., didn't want to raise two waves at once.

cheers, map

18407 Distribution
David H. Crocker,

(J18407) 14-AUG-73 15:00; Title: Author(s): Michael A. Padlipsky/MAP
; Distribution: /DHC ; Sub-Collections: NIC; Clerk: MAP;