Measures, models and measurements for time-shared computer utilities

by G. ESTRIN and
L. KLEINROCK
University of California
Los Angeles, California

INTRODUCTION

There are two major properties of an information processing utility which are forcing intense efforts towards realization despite enormous technological, legal and social difficulties.[1,2] The first of these is the value gained by users from direct access to shared resources made available through the investment of a utility supplier. The second of these is the qualitative change in the value of information banks which may increase in value as a result of use by distributed users. The former property may be realized by distributed resources enhanced by shared centralized resources. The latter property must be realized by integrated shared centralized resources whose efficiency may be enhanced by distributed resources. Neither property may be realized without the design of an effective time-shared processor system.

Recent literature[1-6] has summarized many of the characteristics of time-shared systems which have been or are being built for experimental or commercial application. Fortunately system designers are not waiting for a formal theory before experimenting. Unfortunately some of the results are so diverse in effectiveness that greater emphasis on analysis must be achieved. If predictability cannot lower the risks associated with the tremendous investments required to implement information processing utilities, progress in this field may be markedly decreased.

The present paper presents a characterization of the time-sharing system environment, a set of measures associated with the system, an integrated summary of models and a discussion of measurements. In the final section new proposals for systems, measures, models and measurements are considered.

*This work was supported by the Advanced Research Projects Agency SD-184, the Atomic Energy Commission, Division of Research [AT(11-1) Gen 10] and the Office of Naval Research, Information Systems Branch [Nonr 233(52)].

The time-shared system environment

A time-shared computer system may be viewed as a set of processors, memories and terminals all interconnected by a network of communication channels. Superimposed upon this set of hardware resources is a population of users, a set of shared programs and a set of programmed scheduling algorithms which order the assignment of resources to user requests as they are serviced in the system.

When service is effected according to an a priori scheduling algorithm, the communication network is used to gather user requests arising at *unscheduled* times and to distribute results after servicing ordered queues of requests. Each job is generally taken to completion. When service is effected according to a dynamic scheduling algorithm, processing time quanta, which may be variable, are assigned to each request. When more than one quantum is needed to satisfy a request, a system of queues is used to automatically establish relative queue position based on priorities, quantum requirements and time in-queue. The total processing time *required* for each request is a variable quantity called service time. Processor capability and storage occupancy are variable properties determined by the user requests and assignment and sequencing algorithms incorporated in the service facility. Priorities arriving with user requests can affect both the position in queue and the assignment of resources.

We now consider the scheduling algorithms in a little more detail. A *scheduling algorithm* is a set of decision rules determining which user will next be serviced and how long he will be given use of processing and storage facilities. Thus a newly entering request is placed in a system of queues and, when the scheduling algorithm decides, is given a turn in the processing facility. This turn may or may not be sufficient to satisfy the request. If sufficient the request leaves the system. If not, it reenters the system of queues, leaving a partially com-

pleted program and partially processed data in storage, and waits until the scheduling algorithm decides to give it a second turn, etc.

When a request is given a turn in the processing facility, two other generally dependent scheduling algorithms come into play. One assigns and sequences processors to the request. The second assigns and sequences storage space in the storage heirarchy.

In summary, each user sees a dedicated machine whose response times and charging rates are time varying. The service scheduling algorithm sees a set of unscheduled requests formed into a system of queues as a function of the decision rules, the priority rules, the attained service time and an a priori estimate of service time. The resource scheduling algorithms see a system of queues of processors and storage spaces to be assigned to the flow of entering requests. The objective of the resource scheduling algorithms is to reduce service time. The objective of the service scheduling algorithm is to reduce the overall cost of service; two indicators of the effectiveness are the sizes of the queues and the amount of time any request waits in queue. Choice of one algorithm generally affects performance achieved by the other.

Performance measures

In the context of the environment description of the above Section we may now inquire "What are the qualities which may be considered suitable for definition of performance measures in time shared systems?"

A. The user

Let us define t_I as the time to insert a request at the terminal. This time will be a function of the terminal facilities and the language translators resident in the central facility.

Let us define t_r as the response time of the system, i.e. the time between the receipt by the system of a specified service request and the satisfaction of that request at the terminal.

As indicated in Figure 1 another time called "think time," t_θ is required. "Think time" starts with the response of the system to a request and ends with the insertion of the next request. "Think time" is a function of the user and the same variables affecting t_I.

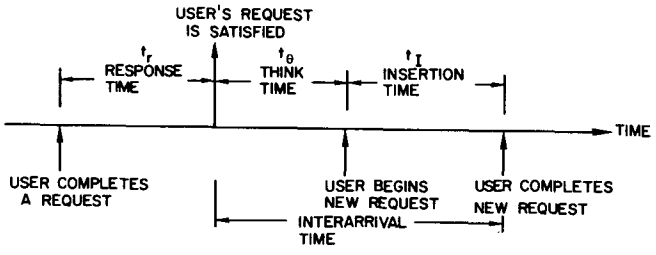

Figure 1—Definition of time intervals.

Let us define C_t as the total cost to a user. This cost function C_t can be defined as a sum of costs of terminal, communication channel, processing, storage and sequencing. When communication costs become sufficiently high it becomes advantageous for the user to establish some more processing power at the terminal.

It is generally agreed that, constrained by cost, the average response time is the single most objective performance measure to the user at present. It is the characteristic which attracts him to the time shared environment. The total time on the terminal is more subjective since it requires adding the characteristics of the user himself. However, it is a quality which must be treated when the user needs to know the probability of meeting a deadline.

B. The service scheduling algorithm

Measures indicating the state of the system of queues include the average waiting time, the average waiting time conditioned on the number of seconds of service (and possibly also on priority group), the average length of queue and the maximum queue length.

Performance measures for the system resources include overhead time, swap time, memory utilization, processor utilization, resident library utilization and channel utilization.

Mathematical models

In this section a number of mathematical models are integrated. They are presented in summary form in Table I. We have broken the models into two groups (infinite population and finite population sources); we have then distinguished between processor-sharing ($Q \rightarrow 0$) and finite Q models where Q is the quantum service interval. In all cases, except as noted, solutions are found for T(t) (the average response time given that the required service time is t) and E (the expected number of users waiting for or in service). Each of the models is considered below.

The usual approach taken in preparing a mathematical model for existing or proposed time-shared service facilities is to treat them as queueing systems. In these models, a user request typically joins some queue, works his way up to the front of the queue, obtains service in the facility for some small amount of time (called a quantum) and then joins the same or some new queue to wait for more quanta if needed. The methods of queueing theory have been applied to a number of such models to obtain various of our measures of performance.

In order to generate and evaluate models of time-shared systems, we must gather data which describe the population of users. Such data then suggest ideali-

Infinite Population Sources						Finite Population Sources					
Q > 0			Q → 0			Q > 0			Q → 0		
Algorithm	Priority-And-Cycle Factor g_{pn}	Reference and Comments	Algorithm	Priority-And-Cycle Factor $\cdot g_{pn}$	Reference and Comments	Algorithm	Priority-And-Cycle Factor g_{pn}	Reference and Comments	Algorithm	Priority-And-Cycle Factor g_{pn}	Reference and Comments
RR	1	[7]	RR	1	[11]	RR	1	[13]	RR	1	[12] Solves for T
RR	g_p	[11]	RR	g_p	[11]				RR	1	[12] Solves for T
FB_n	1	[9]	FB_N= FCFS	1	[9]						
FB ∞	1	[9]	FB ∞	1	[9]						
FB ∞	1	[9] Priority Determines initial queue	FB ∞	1	[9] Priority Determines Loading Point						
FB ∞	g_{pn}	[10] transform of response time obtained									
RR	Function of system state	[17]	RR	Function of system state	[17]						
FB_2	Function of system state	[17]	FB_2	Function of system state	[17]						
Wide Class	g_{pn}	[15] solves for attained service	Wide Class	g_{pn}	[15] solves for attained service	Wide Class	g_{pn}	[15] solves for attained service	Wide Class	g_{pn}	[15] solves for attained service
Wide Class	g_{pn}	[16] states a conservation law	Wide Class	g_{pn}	[16] states a conservation law						
FCFS	—	[18] Bribing Model	FCFS	—	[18] Bribing Model						

Table I: Summary of mathematical queueing models

zations which we may use in our mathematical models and in our simulation models. They also allow us to compare assumed properties of these customers with the measurements.

Following the work of Kleinrock, *et al.* (7-13), we include in the class of queueing systems under consideration, those with the following properties. (See Figure 2.) We assume that the population of new arrivals to the system are separated into P priority groups, this priority being determined by some external property of the arrival (e.g., status in society, wealth, rank, size, memory space required); the assumption here being that the required time in the service facility (e.g., total computation time) is known only to within a probability distribution. Accordingly, let (for p=1,2,...,P),

$A_p(\theta)$ = P_r [customer from p^{th} priority group requires "think time" $\leq\theta$]*

λ_p = average arrival rate of customers to the system from the p^{th} priority group (customers/second)

$B_p(t)$ = P_r [customer from p^{th} priority group requires a total processing time $\leq t$]

$1/\mu_p$ = average service requirement (in operations, say additions, per customer).

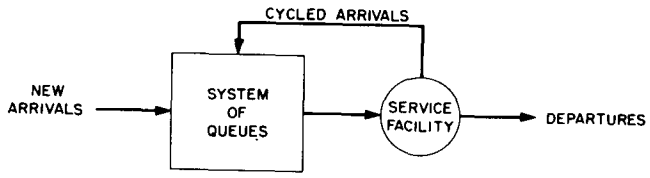

Figure 2—Feedback queueing systems

Upon arrival to the system, a new customer joins some queue in the system of queues. After some appropriate queueing discipline is followed, this customer will then be allowed into the service facility. On his first time through, if he is from priority group p, he will be allotted a maximum of $g_{p1}Q$ second of service where Q is a fixed time interval. If this quantum of time, $g_{p1}Q$, is greater than or equal to his total required service time (t, say) he will then depart from the system as soon as he receives as much time as he needs; if $t > g_{p1}Q$, he will be cycled back to the system of queues where he joins

*See Figure 1 for the definition of "think time". We assume that the insertion time is zero, so that interarrival time and think time are the same.

some appropriate queue and waits for another quantum of service. On his n^{th} visit to the service facility, this customer will receive a maximum of $g_{pn}Q$ seconds of service; thus if

$$Q \sum_{i=1}^{n-1} g_{pi} < t \leq Q \sum_{i=1}^{n} g_{pi}$$

this customer departs from the system during his n^{th} quanta of service.

If $t > \sum_{i=1}^{n} g_{pi}$ he then recycles and continues. g_{pn} will be called the priority-and-cycle factor, and assumed to be greater than zero.

Whenever the service facility ejects a customer (either for departure or re-cycling), some customer (if any are available) is taken into service. This particular customer chosen depends upon the specific discipline used in arranging customers within the system of queues. No customers are allowed to leave before they receive their total required service (i.e., no defecting).

The source which generates the arrivals to the system has itself been modeled in two general ways. One assumption has been to consider that a finite number of consoles is available and that the overall arrival rate to the system is the sum of the individual arrival rates from each of the customers not presently in the system of queues or in the service facility. These systems are referred to as finite population models. A different assumption can be made which considers an infinite population of users, in which case the average think time is taken to be infinite also, so that the average arrival rate of customers to the system is finite and fixed.

Extending from the above we note here that this limit which takes the finite population model into the infinite population model may be properly defined as follows:

We consider the non-priority (P=1) case for simplicity. The average arrival rate ($\lambda \equiv \sum_p \lambda_p$) for a finite number, M, of consoles is merely the average number of consoles E_1 not in queue or in service, times the average arrival rate, γ, per idle console, i.e.

$\lambda = \gamma E_1$. But, for T defined as the average response time

$$E_1 = M \frac{1/\gamma}{T + \frac{1}{\gamma}}$$

since $1/\gamma$ is the average think time. Thus, as $M \to \infty$ and $\gamma \to 0$, we have

$$\lambda = \lim_{\substack{M\to\infty \\ \gamma\to 0}} M\gamma \frac{1/\gamma}{T + \frac{1}{\gamma}} = \lim_{\substack{M\to\infty \\ \gamma\to 0}} M\gamma \qquad (1)$$

since T, the average response time, is finite.

In this same connection, we agree in all these models that the computing facility is not overloaded, i.e., if we define

$$\rho_p = \lambda_p / \mu_p C \qquad (2)$$

and*

$$\rho = \sum_{p=1}^{P} \rho_p \qquad (3)$$

where

P=total number of priority groups

C=capacity of the processor in operations (say additions) per second. Then we insist that $\rho<1$. This insures that the average work load (in operations per second) offered to the processor is less than its capacity to handle such a load.

In all of the following models, the assumption is made that arrival time and processing time of a customer are *independent* random variables; also that these are independent of the values taken on by all other customers. Moreover, we recognize that whenever a customer is moved out of service and another is taken into service, a period of time, called the "swap-time" is required for the transfer. In many of the following models, we assume for the sake of mathematical tractability that this cost, the swap-time, is zero. This assumption certainly should weaken the mathematical results. However, in the models involving a finite quantum, Q, we may think of a portion of that time as being used for swapping; this alters the service time distribution in a predictable way. In the processor-shared models (Q→0), the notion of swapping must be ignored, i.e., we insist that swap-time be zero; this is necessary since we are swapping at an infinite rate and any non-zero swap-time would, by itself, overload the system. Thus, in the zero swap-time models, we obtain results which are idealized in the sense that any non-zero swap-time will only degrade the performance predicted; we recognize that our results are then upper bounds on performance.

We recognize that both the infinite population and the finite population sources are merely special cases of our general model for time-shared systems. Accordingly, we present below the results of analyses of a number of such special cases.

Infinite population models

A number of papers have recently appeared in the literature on time-sharing[7-11] describing models and re-

*Note that ρ_p = average arrival rate of "seconds of work" to the computer from the p^{th} group so ρ is the total average arrival non-saturation. Also, ρ_p is the fraction of time that the enrate from all groups; clearly, this must be less than unity for tire service facility is devoted to processing customers from the p^{th} priority group; also ρ is the probability that the facility is non-empty.

sults for the particular assumptions of infinite population sources. We consider the following.

The round-robin (RR) model[7]

Consider the discrete time model of a time-shared processor studied by Kleinrock.[7] In this model, it is assumed that time is quantized with segments each Q seconds in length. At the end of each time interval, a new unit (or job) arrives in the system with probability λQ (result of a Bernoulli trial); thus, the average number of arrivals per second is λ. The service time (i.e., the required processing time) of a newly arriving unit is chosen independently from a geometric distribution such that for $0 \leq \sigma < 1$

$$s_k = (1-\sigma)\sigma^{k-1} \quad k = 1,2,3,...,$$

where s_k is the probability that a unit's service time is exactly k time intervals long (i.e., that its service time is kQ seconds). Thus, we assume P=1 (no priority distinction among the users) and $1/\mu C = Qk$ (where k is the average of k with respect to the s_k distribution) giving $1/\mu C = Q/(1-\sigma)$. Also $A_p(\theta) = A(\theta)$ and $B_p(t) = B(t)$ are both discontinuous staircase functions given through the binomial and geometric distributions, respectively. Note also that $g_{pn}=1$ for all p and n here. A unit whose processing requirement is kQ will be forced to join the first-come-first-served queue (See Figure 3) k times in all before its service is completed.

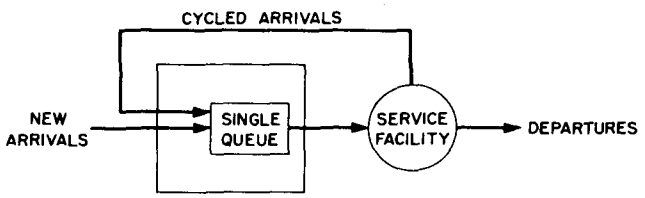

Figure 3—The round-robin system.

For such a system, it has been shown in [7] that the following holds

THEOREM 1: The expected value, T(kQ), of the response time in the round-robin system for a job whose service time is kQ seconds, is

$$T(kQ) = \frac{kQ}{1-\rho} - \frac{\lambda Q^2}{1-\rho}\left[1 + \frac{(1-\sigma a)(1-a^{k-1})}{(1-\sigma)^2 (1-\rho)}\right] \tag{4}$$

where

$$a = \sigma + \lambda Q$$

and

$$\rho = \frac{\lambda Q}{1-\sigma}$$

Furthermore, the expected number, E_r, of customers in the system is given by

$$E_r = \frac{\rho\sigma}{1-\rho} \tag{5}$$

THEOREM 2: The expected value, T'(kQ), of the response time in the strict first-come-first-served system* for a unit whose service time is kQ seconds is

$$T'(kQ) = \frac{QE_r}{1-\sigma} + kQ \tag{6}$$

where E_r is defined in Equation 5.

In reference [7] it is shown that a good approximation to T(kQ) is

$$T(kQ) = kQE_r + kQ. \tag{7}$$

When Equations 6 and 7 are compared we see that for units which require a number of service intervals less (greater) than $1/(1-\sigma)$, the round-robin waiting time is less (greater) than the strict first-come-first-served system. One notes, however, that the average number of service inervals, k, is exactly $1/(1-\sigma)$. Thus, for this approximate solution, the crossover point for waiting time is at the mean number of service intervals.

The case in which $Q \to 0$ is now considered. This corresponds to time-shared systems in which each customer cycles through the system of queues infinitely fast for an infinite number of cycles and spends an infinitesimally small amount of time in the service facility each time he visits it. In a real sense, then, all customers present in the system are using a fraction of the service facility's capacity on a full-time basis. Such an operating procedure may be referred to as a *"processor-shared"* system, and a discussion of its behavior may be found in [11]. The usefulness of this limit of processor-sharing lies in its representation of an idealized sharing operation in which "swap-time" is assumed to be zero. For these cases, the natural analogue of the previous theorem is obtained.

The non-priority processor shared system[11]

This model considers the limit of the round-robin model in which $Q \to 0$ and $\sigma = 1-\mu CQ$, giving a Poisson arrival mechanism with an average of λ units arriving per second and an exponential service distribution with an average of $1/\mu$ operations per customer. This model reduces to a system in which a user is processed at a rate C/K operations per second when there are K users sharing a computer of capacity C. This processing rate varies as new users enter and old ones leave the system. A harmonic variation of individual processing rate with number of customers is assumed.

THEOREM 3: The expected value, $T(l/C)$,

*This is a reference system and corresponds to the more usual case where a unit receives its complete processing requirement the first time it enters service.

of the response time in the processor-shared system for a customer requiring l operations, is

$$T(l/C) = \frac{l/C}{1-\rho} \tag{8}$$

where

$$\rho = \lambda / \mu C.$$

The expected number, E, of customers in the system is

$$E = \frac{\rho}{1-\rho} \tag{9}$$

The priority processor-shared system[11]

In this system, there are P priority groups with Poisson arrivals at an average rate of λ_p per second and an exponentially distributed service requirement with a mean of $1/\mu_p$ operations (p=1,2, ...,P). Assign a customer from the p^{th} priority group a capacity f_pC when there are n_i type i customers in the system; f_p is given by

$$f_p = \frac{g_p}{\sum_{i=1}^{P} g_i n_i} \tag{10}$$

For such a system,

THEOREM 4: The expected value $T_p(l/C)$ of the response time spent in the priority processor-shared system for a customer from priority group p who requires l operations is

$$T_p(l/C) = \frac{l}{C}\left[1 + \sum_{i=1}^{P} \frac{g_i \rho_i}{g_p(1-\rho)}\right] \tag{11}$$

the expected number, E_p, of type p customers in the system is

$$E_p = \frac{\rho_p}{1-\rho}\left[1 + \sum_{i=1}^{P}\left(\frac{g_i}{g_p} - 1\right)\rho_i\right] \tag{12}$$

First-come-first-served model (FCFS)[11]

For completeness, consider a strict first-come-first-served system with the same input and service requirements as in our priority model. To this end,

THEOREM 5: The first-come-first-served system with a priority input yields, for customers with l required operations, an expected response time as follows:

$$T(l/C) = \frac{l}{C} + \frac{\rho/\mu C}{1-\rho} \tag{13}$$

where

$$\frac{1}{\mu C} = \frac{\rho}{\sum_{p=1}^{P} \lambda_p}$$

Note that P=1 yields the (non-priority) processor-shared system.

The multiple level processor-shared model[8-10]

This model, which is denoted by FB_N where N is the number of levels, is shown in Figure 4. Make the assumptions of exponential interarrival and service times. A unit at the service point at any given queue level will not be serviced unless all lower numbered queues are empty. Thus, immediately after a unit has received service the next unit serviced will be the one at the service point of the lowest level, non-empty queue. This unit will be given a quantum Q of service as in the round-robin model; if more is needed then the unit is subsequently placed at the end of the *next higher* level queue, otherwise it leaves the system.

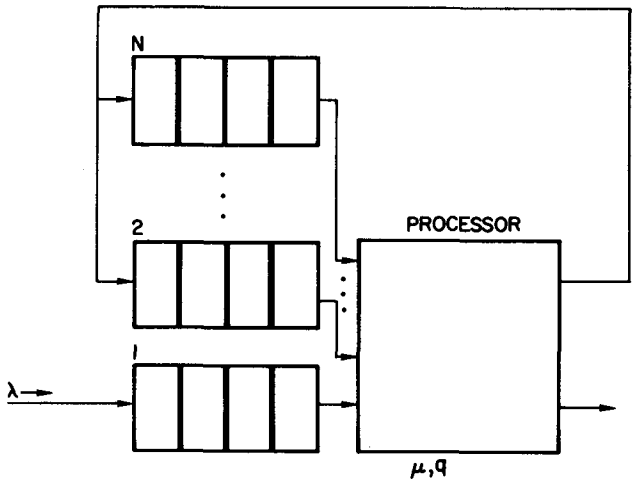

Figure 4—The FB_N model.

For $N<\infty$, assume that the N^{th} level queue is a quantum-controlled, first - come - first - served (FCFS) queue. Specifically, units at the N^{th} level are served a quantum at a time until completion (i.e., there is no round-robin in the N^{th} queue but an arrival to a lower level during the servicing of an N^{th} level unit will preempt this unit after it has completed the quantum-service in progress). Note that, with these assumptions, FB_1 denotes the conventional FCFS system.

The average response time (T)t has been obtained for Q>0 (see [9]). Below are given results for the processor-sharing limit, Q→0. For finite N the FB_N system reduces to a FCFS system. Of greater interest is the limiting case Q=0 when we assume N=∞. By arguments based on very small Q sizes it can be seen that the resulting system can be viewed as corresponding to a system in which arrivals always preempt the unit, if any, in service and are allowed service until their service time exceeds that having been received by some other unit in the queue. We have the following theorem from [9].

THEOREM 6: The average response time for customers requiring t seconds of service in the processor-shared FB_N system is

$$T(t) = \begin{cases} \dfrac{1}{1-\rho}\ (1/\mu); & N < \infty \\ \dfrac{(\lambda/2)\int_0^t x^2 dF(x)}{\left[1-\rho(1-\epsilon^{-\mu t})\right]^2} + \dfrac{t}{1-\rho(1-\epsilon^{-\mu t})}; & N = \infty \end{cases} \quad (14)$$

where,

$$F(x) = \begin{cases} 0; & X < 0 \\ 1-\epsilon^{-\mu x}; & 0 \leq x < t \\ 1; & x \geq t \end{cases} \quad (15)$$

Coffman [8] has also considered such systems with externally assigned priorities. These priorities determine the initial queue which a customer joins upon entry to the FB_N system (the lower priority units join higher numbered, lower priority queues). He also carries this extension to the processor-shared case.

Schrage [10] has also considered such systems. He has generalized the quanta to depend upon n, i.e., $g_{pn} = g_n$. Also he allows an arbitrary service distribution B(t). However, he restricts his investigation only to the infinite level case, $N=\infty$. Schrage solves for the Laplace transform of the response time, and this allows him to obtain the moments of this measure (in particular, he obtains the mean and variance).

Finite population models

The principal model in this category has been studied by Scherr [12]. He assumes that think time and service are both independently and exponentially distributed. A total of M consoles is assumed to be available and a non-priority processor shared system is considered, i.e., $Q\to 0$, $g_{pn}=1$. His main result, for a single processor is

$$T = \frac{M/\mu C}{1-\pi_0} - \frac{1}{\gamma} \quad (16)$$

where $\pi_0 = P_r$ [no customers are receiving or awaiting service]

$$= \left[\sum_{j=0}^{M} \frac{M!}{(M-j)!}\left(\frac{\gamma}{\mu C}\right)^j\right]^{-1} \quad (17)$$

and

$$\frac{1}{\gamma} = \text{avg. think time}$$

We note here that as $M\to\infty$, $\gamma\to 0$, such that $M\gamma=\lambda$, we arrive at the infinite population model of a non-priority processor shared system as follows:

$$\lim_{\substack{M\to\infty \\ \gamma\to 0}} \pi_0 = (1-\rho) \lim_{\substack{M\to\infty \\ \gamma\to 0}} \left[1 - \frac{\rho\gamma/\mu C}{(1-\rho)^2}\right]^{-1} \quad (18)$$

Applying this last to Eq. (16) we get

$$T = \frac{1/\mu c}{1-\rho} \quad (19)$$

Inspection of Eq. (8) reveals that the average response time for all jobs (averaged over service requirements) yields exactly T as above. Furthermore, in the limit, it is easy to show that

$$E = \rho/(1-\rho)$$

as in Eq. (9). This confirms the agreement between the two models in the limit.

Scherr also derives a similar expression for T in the case where the processing faciltity consists of more than one processor.

Greenberger [13] has considered a finite population model with a round-robin scheduling algorithm under exponential assumptions of think time and service time with a finite quantum Q. In addition, he includes a swap-time of V seconds. He obtains an approximation which can be used to obtain the response time conditioned on the service time kQ as

$$T(kQ) = \frac{k}{\mu C}\left(1-e^{-\mu CQ} + \mu CV\right)\left[\frac{M}{1-\pi_0'} - \frac{(\mu C)'}{\gamma}\right] + \left(\frac{1-\pi_0'\left(1+\frac{M\gamma}{(\mu c)'}\right)}{1-\pi_0'}\right)\left(\frac{S_2-V^2}{2(S_1+V)} - S_1\right) \quad (20)$$

where

$$\frac{1}{(\mu C)'} = \frac{1}{\mu C} + \frac{V}{1-e^{-\mu CQ}}$$

$$S_1 = \frac{1}{\mu C}\left(1-e^{-\mu CQ}\right)$$

$$S_2 = \frac{2}{\mu C}\left(S_1 - Qe^{-\mu CQ}\right)$$

and π_0' is the expression for π_0 given in Eq. (17) with μC replaced by $(\mu C)'$. A similar model has also been studied by Patel [14]. In Eq. (20) as $V\to 0$, $Q\to 0$, we note that T(k Q) averaged over k reduces to the ex-

pression in Eq. (16) given by Scherr for the processor-shared systems. The same comment applies when one takes this to the limit of an infinite population system, obtaining Eq. (19) above.

Additional results

The distribution of attained service has been defined as (see [15])

$N_p(\tau)$ = expectation of the number of customers in the system of queues and in service from priority group p who have so far received exactly τ seconds of service.

This quantity, $N_p(\tau)$, gives one a description of the composition of the various queues, and a measure of the relative state of partial service received by those customers still in the system. The results for attained service are very general.

There exists [16] a conservation law which states that an appropriate average of the mean response times (by priority) can be *invariant* to scheduling algorithm.

Coffman [17] has considered models (for $Q>0$ and $Q\to 0$) with variable quanta for both RR and FB_2 in which a customer in service receives an additional quantum for each new customer who enters while he is in service. This tends to reduce the swap-time.

The notion of paying a price for one's priority is not an unfamiliar one—it is often referred to as bribing [18]. A bribing model has been considered in which customers offer a bribe (based upon an "impatience factor" of their own) to gain preferred position on line. All those bribing strategies are then characterized which minimize an appropriately defined cost function averaged over the population of users.

When we consider more than a single processor, a number of possibilities present themselves. We may choose to use all processors in parallel, with or without constraints (see Coffman [8]) or in series (see Kleinrock [19]) or in some series-parallel combination (see [20-21]). The problem associated with providing the interconnection of a network of processors is a major one; some work along these lines may be found in [21,22].

Measurements and simulation

Measurements made on actual time-shared systems or on simulated models of time-shared systems have the following purposes:

1. To make a choice between one system and another.
2. To deepen the understanding of a system so as to affect the design of other systems or to optimize a given system.

In the first case the user is interested in measurements of response time and both the cost and quality of the system as discussed in the second section. In the second case the model builder is interested in measurements of those parameters which arise in his model and which can be subjected to reasonable measurement. Examples of the latter measurements are the average number of active consoles, queue size, waiting time in queue, response time, service time and think time. In the second case the system designer is interested in measurements which are easy to make and which most directly lead to optimization techniques as well as support the building of models. These last measurements will therefore be concerned with the efficiency of processor and storage allocation in addition to the scheduling algorithm affecting the input queue.

Below we seek to integrate a number of simulation and measurement results which have appeared in the literature. Table II summarizes reported measurements of user characteristics and Table III summarizes reported measurements of performance characteristics. In both cases references are denoted with script indicating simulation and bold face indicating actual system measurements.

In similar fashion to section 3, essential characteristics of a number of experiments and then the measurement results are discussed below.

Table II: Data on User Characteristics

User Characteristic	Reference
Probability Distribution of Think Time	S,T
Probability Distribution of Service Time	C,S,T
Probability Distribution of Program Size	C,S,T
Priority vs. Program Size	SD
Percentage User Time vs. Day	SD
Probability Distribution of Disc Requests	C

Table III: Data on performance characteristics

Performance Characteristic	Reference
Response Time/Processor Time vs. M	S,T,*S*
Response Time vs. Processor Time	*S*
Response Time vs. M	*T*,R
Probability Distribution of Response Time	*S*,L
$\bar{M}$ vs. M	S,*S*
Probability Distribution of M	T,*TR*
$\bar{M}$ vs. Q	*T*M
Computer Utilization vs. M	S,*S*,*TR*
Computer Utilization vs. Q	*TM*
Breakdown of Computer Utilization vs. Processor Time	C
Batch Processing RR Computer Utilization Change	T,*TM*
Tape Usage vs. M	T
Utilization of Swap Storage vs. M	S
User, Idle, and Disc Time vs. Day	SD
Breakdown of Computer Time Usage	S
Memory Map vs. Program Size	SD

The experiments

We consider the results of 4 simulation efforts and 4 measurement efforts. Among the simulations, we have:

1. Scherr (*S*) [12] has simulated the M.I.T. Project MAC's Compatible Time-Shared System (C.T.S.S.—see Corbato [23]). The C.T.S.S. scheduling algorithm is a variation on the FB_9 system described earlier with $g_{pn} = 2^{n-1}$ and Q=0.5 seconds; the variation includes a priority system in which program length is used to place new programs onto the various queues (longer length receiving lower priority). He considers these simulation models: CTSS; CTSS with RR ($g_{pn} = 1$, Q = 2 seconds) instead of FB_9; and CTSS with multiprogramming to provide overlapped processing and swapping.

2. Fine and McIsaac (F.*M.*) have simulated a system similar to the S.D.C. Time-Sharing System [24]. They consider an RR system as well as a two-queue scheduling procedure (similar to FB_2). The second (lower priority) queue contains customers who have submitted production jobs and who are serviced to completion (first-come-first-served) during the idle intervals between "non-production" jobs which are served RR on the first queue (high priority).

3. Fife and Rosenberg (F.*R.*) [25] have simulated a system in which the central notion is one of memory-sharing with fixed memory partitioning. The main core is divided into 5 memory blocks. Each user is confined to remain within a single block. Jobs queue up for an allocation to core memory, and, once in core, remain there until the job is completed or until the program halts awaiting an operator message (or until an illegal command is generated). Processing is transferred to a new job in core when the current program being run fills its output buffer.

4. Lavita (*L*) [26] has simulated a first-come-first-serve run-to-completion system with emphasis on the comparison between a drum system and a disk system.

Among the actual system measurement efforts, we include the following:

1. Scherr (S) [1] has made measurements of the behavior and system performance on the MIT Project MAC's CTSS [23] as described earlier. During the period of his measurements, CTSS served approximately 250 users with an IBM 7094 (augmented with disc and drum storage) connected through an IBM 7750 transmission controller to users' remote consoles (each consisting of a keyboard and printer). The core memory consisted of 2 modules, each of 32,768 36-bit words, with an access time of 2 microseconds (the first module held only the supervisory program and the other was used for users' programs during execution).

2. Totschek (T) [27] reports measurements made on the SDC Time-Sharing System [28] which can handle up to 35 users at teletypes at remote consoles. Inputs from the user consoles pass through a D.E.C. PDP-1 computer to the central processor. The central processor is an IBM manufactured AN/FSQ-32, which contains 65,536 48-bit words of high speed memory (of which 48,784 are available for user programs) with a cycle time of 2.5 microseconds. A high speed buffer of 16,384 words is the interface between the AN/FSQ-32 and the PDP-1. Two scheduling algorithms are used. The first is an RR scheme (with $g_{pn}=1$, Q=400 ms). The second is an FB_2 system where $g_{p1}=1, g_{p2}=50$; the second queue is automatically interrupted (pre-empted) when the first queue becomes non-empty.

3. Sutherland (SD) [29] has made measurements on the Lawrence Radiation Laboratory (LRL) Time Shared System which is implemented on the CDC 6600. The system handles 48 teletype-equipped consoles which serves 110 users. Each console connects to one of ten peripheral processors (each containing 4096 12-bit words). One other of these peripheral processors carries out the scheduling task. The mass memory is a disk. Central memory consists of 30,000 words reserved for supervisory tables and 100,000 words available to the user. Up to 10 users may be in this second section of memory at any one time. The scheduling algorithm among these 10 is RR priority system with $g_{pn}=p$, Q=40 ms. Loading of new programs is done on a priority as well as an available memory space basis. Every 3¾ seconds the system does a priority load which removes a lower priority job if, by so doing, a higher priority job can be loaded and run. This is subject to the guarantee that a program will not be dumped for a number of microseconds equal to 128p times the program size(in code words). Each user is charged (against his daily allotment) an amount of time equal to his central processor time (CPU) plus his input-output time (I/O) times his priority p, i.e. (CPU+I/O)p minutes. A user's priority is self-assigned!

4. Cantrell (C) [30] has reported on his measurements of the GE-Dartmouth Time-Shared System. This system handles 200 users at remote teletype terminals. Inputs from user terminals are processed by GE D-30's en route to the central processor. Two D-30's are at Dartmouth for their users. One D-30 is at Boston and one D-30 is at New York City for GE users. The central processor consists of one GE 635 with 65,536 36-bit words of high speed memory (of which 25K are available for user programs) with a cycle time of 2 microseconds. The memory heirarchy consists of two Disks (16 M each), Drum (50,000 words), 6 Tape Drives and an RCA RACE file (340 M characters). The system uses a simple round robin scheduling algo-

rithm. Each user gets a fixed 200 milli-second quantum.

Results of simulation and measurement

The data obtained falls into two categories—user characteristics and performance characteristics; we list these obtained data in Table II and Table III respectively. We define $\bar{M}$ as the number of active consoles* and M as the average value of M. The characteristics listed in these tables refer to graphs, figures, or tables in the references cited.

Of primary interest with respect to user characteristics is the distribution of interarrival times (or think times). One finds a fair degree of agreement between the two sources of measurement, and it is interesting to note that both yield an average think time of approximately 35 seconds.* In these two cases, the standard deviation of think time was the same order of magnitude as the average. Both sources show curves which seem to indicate that the think times are approximately exponentially distributed.

The other major user characteristic is the distribution of service time (swap time excluded). The three sets of data agree to some extent here, all ranging over several orders of magnitude with slowly decreasing tails. The median falls around 0.05 seconds, with much larger variances. The mean values do not agree too well, but this variation is understandable due to the great sensitivity of the mean to the details at the tail of the distribution. Totschek comments that the log-normal distribution fits both service time and think time reasonably well. It seems that an hyperexponential distribution would do well also. Cantrell observes that the shortest 90% of all service times represents just slightly more than 10% of the system load (in processor time) and, therefore, that the longest 10% account for almost 90% of the load (compute time)!

The distribution of program sizes provides little agreement among the sources. Scherr reports an average program size of 6.3×10^3 words.

A thorough discussion of the wealth of information available on performance characteristics listed in Table III can provide a basis for many future papers. Rich as these data are, a great deal remains to be done, not only in making additional measurements, but also in determining more compact measures of time-shared system performance. We choose to discuss only certain highlights of the data in Table III.

One of the most significant observations one can make is with respect to the first item in Table III. The results given by Scherr for the ratio of response time to processor time as a function of M for the CTSS measured data, the CTSS simulation and the RR simulation all agree remarkably well. Furthermore, they all agree with his RR processor-shared model! Totschek's curves are similar in shape to Scherr's, but an absolute comparison is difficult to make because Totschek includes I/O time in the response time.

A quantity which is difficult to solve for analytically from the mathematical models is the probability density function of the response time (usually only its mean value, T, is found). This density can be obtained through simulation, however, and both Scherr and Lavita have done this. In the CTSS and RR cases, Scherr obtains a function which looks like an exponential distribution. For a first-come-first-served algorithm, Lavita obtains a distinctly different distribution, which shows very few short response times; this is easily seen to be due to the relatively long waits that even short jobs must experience due to the run-to-completion rule.

The remaining measurements concerned with system performance are widely dispersed in character and of a probing nature rather than strongly relevant to some model. They reflect the observations of Estrin, *et al.*[31] that present methods of measurement do not allow sufficient freedom in design of experiments on complex systems.

Now let us consider the relevancy of the tabulated measurements to the previously discussed models and vice versa.

The infinite population model cannot be used to predict the characteristics of a finite console system of the types measured. These systems inherently reduce the arrival rate of requests as the number of busy consoles approaches the total number of consoles, i.e. they are self-correcting. However, in the case of a large information processing utility we may well see a system with a huge number of terminals most of which are idle at any one time. In such a case the infinite population model assumptions would hold. This would be the case $M \to \infty$, $\gamma \to 0$ discussed in the third section. Therefore it is apparent from Table I that most of the work on modeling has been dealing only with a futuristic condition.

In the case of the finite population models it can be observed that:

1. Scherr has demonstrated predictability of his model. It might be of interest to explore the effectiveness of Scherr's adjustment for swap-time under conditions of varying Q.
2. In the case of Greenberger's model, which includes the effect of swap time under assumptions of round robin and exponential service time and think time, some computation could explore the predictability.

*This reflects the load on the system.

*In fact, Scherr found this mean value to vary only slightly with time of day, day of week, and number of active consoles.

3. In the multiprocessor case both models and simulated or constructed systems are too sparse at the present time to expect many results.

SUMMARY

The analysis of existing measures, models and measurements in the previous sections reveals a number of aspects which should be considered in future studies.

All presently proposed time-shared computer systems consider scheduling algorithms which favor the short problems. It seems apparent that any information processing utility which is to be established must deal with an environment which attracts all large customer groups. Hence we should consider for modelling and measurements a time-shared multi-processor system which has the following properties:

A. Multiple resources are established such that Resource $_i$ favors requests in the i^{th} interval of a distribution of problem lengths.
B. An incentive is provided for the user to predict correctly that his request will be satisfied best by Resource$_i$.
C. The system automatically moves user requests from one resource to another if the requests are not satisfied by a prescribed number of units of service. A penalty is incurred by the user for such overhead.
D. The system automatically moves user requests from one resource to another to relieve congestion.
E. Some memories are shared by the multiple resources.

The weakness of the infinite population terminal models in even being able to make predictions about practical finite population systems has been strongly stated. On the other hand the relatively small number of studies of finite population models must be rectified. In particular it would be desirable to extend those models to provide predictions of response time conditioned on service time required. This is important in order to predict how much short problems are favored and how much long problems are disfavored. Another aspect arises in consideration of the success of Scherr's model in predicting measurements after an empirical adjustment for swap time. There is a need for models of resources and their allocation so that we may investigate improvement in swap time and other overhead factors. Scherr[32] has demonstrated a reasonably precise definition of saturation and indicates that simulation may be used to predict performance of proposed time-shared systems. Aside from our need for the possible generation of other system saturation models, this again raises the need for improved models of system resources to be used in simulation or analytical studies.

Most of the weakness in the relevancy of measurements to models arises, at present, from the unavailability of tools for making desired observations of dynamic systems. The recent concentration of some studies[31] on the use of a computer to observe others as well as the identification of methods for useful self measurement gives some hope of progress in this area. This problem is particularly acute with respect to optimization of resource allocation since simulation procedures can become overly costly when such system detail is exposed.

The optimization of given systems is somewhat hampered by the unavailability of statistically significant test environments. If it were possible to capture records of requests to and responses from operating time-shared systems over a sufficiently long time span, they might provide test environment inputs to a system under study. It is true that such test conditions ignore the adaptability of the population of users but such static "tuning" of time-shared systems might provide greater insight through reproducible experiments.

REFERENCES

1 M R IRWIN
The computer utility
Datamation 12 22-27 1966

2 M HYMAN
The time-sharing business
Datamation 13 49-57 1967

3 *Symposium on computers and communication, their system interaction*
Sponsored by the IEEE Communications Technology Group and the Computer Group
January 1967 in Santa Monica, California (unpublished)

4 G ESTRIN
The information processing public utility
ACM Lecture 1967 (unpublished)

5 C V PARKHILL
The challenge of the computer utility
Addison-Wesley Mass. 1966

6 R M FANO
The MAC system: the computer utility approach
IEEE Spectrum 56-64 1965

7 L KLEINROCK
Analysis of a time-shared processor
Naval Research Logistics Quarterly 11 59-73 1964

8 E G COFFMAN
Stochastic models of multiple and time-shared computer operations
UCLA Department of Engineering Report No. 66-38 June 1966

9 E G COFFMAN and L KLEINROCK
Some feedback queueing models for time-shared systems
(to be published)

10 L E SCHRAGE
The queue M/G/1 with feedback to lower priority queues
Management Science 13 466-474 1967

11 L KLEINROCK
Time-shared systems: a theoretical treatment
JACM 14 242-261 1967

12 A L SCHERR
An analysis of time-shared computer system
M.I.T. Project MAC report MAC-TR-18 (Thesis) June 1965
13 M GREENBERGER
The priority problem and computer time-sharing
Management Science 12 888-906 1966
14 N R PATEL
A mathematical analysis of computer time-sharing systems
M.S. Thesis Department of Electrical Engineering M.I.T. Cambridge Mass May 1964
15 L KLEINROCK
Distribution of attained service in time-shared systems
Presented at Workshop on Models for Time-Shared Processing in Symposium in Ref 3
16 L KLEINROCK
A conservation law for a wide class of queueing disciplines
Naval Research Logistics Quarterly 12 181-192 1965
17 E G COFFMAN
Analysis of time-sharing algorithms designed for limited swapping
Presented at Workshop on Models for Time-Shared Processing in Symposium in Ref 3
18 L KLEINROCK
Optimum bribing for queue position
To appear in Operations Research
19 L KLEINROCK
Sequential processing machines (S.P.M.) analyzed with a queueing theory model
JACM 13 179-193 1966
20 B RUSSELL
Properties of a variable structure computer system in the solution of parabolic partial differential equations
Dissertation (Ph.D.) Department of Engineering University of California Los Angeles September 1962
21 L R FORD JR and D R FULKERSON
Flows in networks
Princeton University Press New Jersey 1962
22 L KLEINROCK
Communication nets; stochastic message flow and delay
McGraw-Hill New York 1964
23 F J CORBATO *el al.*
The compatible time-sharing system
M.I.T. Press Cambridge 1962
24 G H FINE and P V McISSAC
Simulation of a time-sharing system
SDC Report SP-1909 December 29 1964
25 D FIFE and R ROSENBERG
On queueing in a memory shared computer
Report from Cooley Electronic Laboratory University of Michigan Ann Arbor
26 P P LAVITA
Digital simulation of a time-shared system
M.S. Thesis in Mathematics New York University February 1966
27 R A TOTSCHEK
An empirical investigation into the behavior of the SDC time-sharing system
SDC Report SP-2191 August 27 1965
28 E G COFFMAN, J I SCHWARTZ and C WEISSMAN
A general-purpose time-sharing system
Proceedings of the Spring Joint Computer Conference Spartan Books Washington D.C. 1964
29 G G SUTHERLAND
Talk presented at Workshop on Models for Time-Shared Processing in Symposium in Ref 3
30 H CANTRELL
Talk presented at Workshop on Models for Time-Shared Processing in Symposium in Ref. 3
31 G ESTRIN, D HOPKINS, B COGGAN and S CROCKER
Snuper computer—a computer instrumentation automaton
AFIPS Conference Proceedings Vol 30 645-656 Thompson Book Co Washington D.C. 1967
32 A L SCHERR
Time sharing measurement
Datamation 22-26 April 1966