A study of time sharing systems considered as queueing networks of exponential servers

L. Lipsky*

Computer Graphics Group, Informatica, Katholieke Universiteit, Nijmegen, the Netherlands

The queueing theory of exponential servers as applied to time sharing computer systems is discussed in detail with respect to response time and throughput. The restriction that only a fixed number of transactions may be processed simultaneously by the computer subsystem is treated, with comparison made among four different approximations.

(Received September 1978; revised October 1979)

1. Introduction

There have been many studies made of the throughput of computer systems using what is by now a standard set of assumptions of queueing theory (Bhandiwad and Williams, 1974; Buzen, 1973; Chandy *et al*, 1975; Cox and Smith, 1961; Gordon and Newell, 1967; Jackson, 1963; Kleinrock, 1975; Lipsky, 1974), namely that all servers have exponentially distributed service times, and the steady state solutions apply. There has been some success in this approach (Boyse and Warn, 1975; Brandwajn, 1974; Chiu *et al*, 1975; Giammo, 1976; Kleinrock, 1976; Lipsky, 1974; Lipsky *et al*, 1977; Lipsky and Church, 1977) and several attempts have been made to apply this technique to 'time sharing systems', where the computer is not a closed network, but rather is opened to a so-called 'thinking' stage (Chandy *et al*, 1975; Kleinrock, 1976; Scherr, 1967). Such models have been less successful at estimating response time versus active terminals, for instance, because the models are very sensitive to certain unknown or poorly known parameters.

Our purpose here is to examine and compare several methods for dealing with time sharing systems and see what conclusions can be drawn therefrom. It is sometimes useful to have an illustrative example, so when appropriate, we will use parameters pertaining to the PDP11/45 minicomputer running under 'UNIX' which is operated by the Computer Graphics Group at Nijmegen University, the Netherlands (Lipsky, 1977).

2. Theoretical background

It is assumed here that the reader is familiar with any one of several articles where the theory is expounded in detail. We only summarise here. It has been shown by Jackson (1963) and by Gordon and Newell (1967) (see also Buzen, 1973) that for any network of M service stages with exponentially distributed service times, and serving a total of N customers the probability of being in the state $\bar{n} = \{n_1, n_2, \ldots, n_M\}$ where $n_1 + n_2 \ldots + n_M = N$ is

$$P(\bar{n}) = \frac{X_1^{n_1}}{\beta_1(n_1)} \times \frac{X_2^{n_2}}{\beta_2(n_2)} \ldots \times \frac{X_M^{n_M}}{\beta_M(n_M)} \Big/ G(N) \qquad (1a)$$

n_i is the number of customers at stage 'i', and the X's can be defined to within an arbitrary multiplicative constant, as the fraction of time a customer spends at each of the stages when there is no one else in the system. The function, $G(N)$, normalises the probabilities so that their sum is 1, so

$$G(N) = \sum_{\substack{\text{All} \\ \bar{n}}} \prod_{i=1}^{M} \frac{X_i^{n_i}}{\beta_i(n_i)} \qquad (1b)$$

The load functions, $\beta_i(n)$ are defined by:

$$\beta_i(n) = \alpha_i(1) \times \alpha_i(2) \ldots \times \alpha_i(n) \qquad (2)$$

where the mean service rate when there are n customers at the i^{th} stage is $\alpha_i(n)/t_i$. By definition, $\alpha_i(1) = 1$. A customer may return many times to the i^{th} stage, and t_i represents the mean time it takes to serve him for each visit. When $\beta_i(n) = 1$ for all n, the i^{th} stage is 'load independent'.

There is one class of load-dependent stages which should be mentioned. Consider a stage with one queue emptying into several ($r \geqslant 2$) identical servers. The probability of a completion in a small interval of time Δt if there is only one customer there is simply $\Delta t/t$, where t = mean service time of one server.† However, if there are two customers in the queue, then both are being served and the probability of a completion becomes $2\Delta t/t$. In general,

$$\begin{aligned} \text{Probability of completion} &= n\Delta t/t \text{ for } n \leqslant r \\ &= r\Delta t/t \text{ for } n \geqslant r \end{aligned} \qquad (3a)$$

In this case,

$$\begin{aligned} \beta(n) &= n! && \text{for } n \leqslant r \\ &= r!\, r^{n-r} && \text{for } n \geqslant r \end{aligned} \qquad (3b)$$

If a situation occurs where n can never exceed r then $\beta(n) = n!$ always. This is the case in a time sharing system, where the banks of available terminals can be considered to be a single stage, and the number of customers (programmers using terminals) can never exceed the number of terminals. A given server (terminal) is 'active' while the customer is preparing his request to the computer system (thinking), and is inactive while waiting for the computer's response. The computer load in responding to a request is called a 'transaction'.

Several general statements can be made concerning properties relating to equations (1). First, it can be shown that

$P_i(n_i, N)$ = Probability of n_i customers being at queue i with a total of N in the system

$$= \frac{X_i^{n_i}}{\beta_i(n_i)} g_i(N - n_i)/G(N) \qquad (4a)$$

where the function, g_i is defined by:

$$g_i(n) = \sum_{\text{all}} \frac{X_1^{n_1}}{\beta_1(n_1)} \ldots \times \frac{X_{i-1}^{n_{i-1}}}{\beta_{i-1}(n_{i-1})} \times \frac{X_{i+1}^{n_{i+1}}}{\beta_{i+1}(n_{i+1})} \ldots \frac{X_M^{n_M}}{\beta_M(n_M)} \qquad (4b)$$

and the sum is over all possible sets of n_i such that

$$\sum_{j \neq i}^{M} n_j = n \leqslant N, \quad \text{and } n_j \geqslant 0 \qquad (4c)$$

Furthermore,

*Permanent Address: Department of Computer Science, University of Nebraska, Lincoln, Nebraska 68588, USA.

†When, as here, we are discussing a single stage and need not distinguish between different stages, the subscript 'i' is dropped from β_i, t_i, etc.

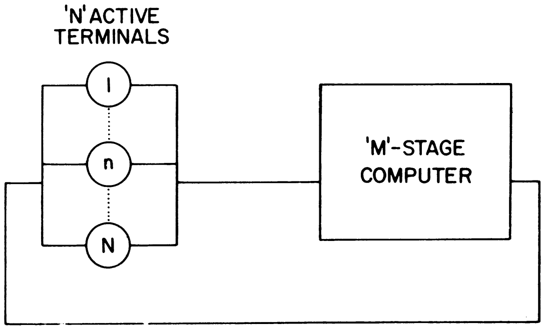

Fig. 1 Diagram of a Terminal-Computer System. The computer is of arbitrary complexity, having 'M' distinct devices, and there are N terminals 'logged on' and active

$$G(N) = \sum_{n=0}^{N} \frac{X_i^n}{\beta_i(n)} g_i(N-n) \cdot \tag{5a}$$

This is a special case of a more general statement which allows (1b) to be written in the form:

$$G(n) = \sum_{n=0}^{N} g_1(n) g_2(N-n) \tag{5b}$$

where g_1 and g_2 are normalising functions for any pair of disjoint subsystems whose union is the entire system. That is, they are of the same form as G, in equation (1b), each including only those X_i's which correspond to servers in its subsystem.

It can be shown (Williams and Bhandiwad, 1976) that the mean service rate of stage i with arbitrary load dependence is simply

$$\lambda_i(N) = \frac{X_i}{t_i} \times G(N-1)/G(N) \tag{6}$$

where $\lambda_i(N)$ is the mean number of customers passing through stage i per unit time when there are N customers in the entire system. In the case when the i^{th} server is load independent ($\beta_i(n) = 1$), we also have

$$\lambda_i(N) t_i = \text{probability that the server is active,} \tag{7a}$$

while for $\beta_i(n) = n!$,

$$\lambda_i(N) t_i = \text{mean number at stage } i\,. \tag{7b}$$

Equations (7) follow directly from Little's Law (Kleinrock, 1975), which states that the mean arrival rate of customers to a subsystem multiplied by the mean time spent by a customer in or at that subsystem is equal to the mean number of customers there ($\bar{n} = \bar{\lambda} t$).

3. Time sharing model

Consider a system according to **Fig. 1.** The M-stage computer can be of arbitrary complexity, but from which a generating function $g(n)$ can be calculated according to equation (4b) for $i = 0$. From equation (5a)

$$G(N) = \sum_{n=0}^{N} \frac{Z^n}{n!} g(N-n) = \sum_{n=0}^{N} \frac{Z^{N-n}}{(N-n)!} g(n) \tag{8a}$$

where Z = mean time a customer takes to submit the next request to the computer after receiving his previous response (so called 'think time'). The X_i's for the computer subsystem (the parameters needed to calculate $g(n)$) have been normalised to be the mean times spent by each stage in satisfying a single transaction. The expression:

$$W(N-n) = \frac{Z^n}{n!} \frac{g(N-n)}{G(N)} \tag{8b}$$

represents the probability of there being n active transactions, and $N - n$ customers thinking when there are N in the system.

The mean number thinking at any time (from equations (6) and (7b) with $i = 0$) is:

$$\bar{N}_0(N) = Z\, G(N-1)/G(N)\,, \tag{9a}$$

owing to the specific normalisation selected, where $Z = X_0 = t_0$, and the mean number of transactions processed per unit time is

$$\lambda(N) = G(N-1)/G(N)\,. \tag{9b}$$

The response time $R(N)$ (the time a transaction spends in the computer subsystem) must be by Little's Law

$$R(N) = \frac{\text{Number in Subnetwork}}{\lambda(N)} = \frac{N - \bar{N}_0}{\lambda(N)} = \frac{N}{\lambda(N)} - Z\,. \tag{10}$$

Observe that R grows with N, and for Large N,

$$R(N) = \frac{N}{\lambda(\infty)} - Z + 0\left(\frac{1}{N}\right) \tag{11}$$

$\lambda(\infty)$ is itself determined by the maximal capacity of the computer and is independent of Z

$$\lim_{N\to\infty} \frac{1}{\lambda(N)} = \lim_{N\to\infty} \frac{G(N)}{G(N-1)} = \lim_{N\to\infty} \operatorname{Max}_{i=1}^{M} \left[\frac{X_i}{\alpha_i(N)}\right] \tag{12a}$$

provided the $\alpha_i(N)$'s have a limit greater than zero. This limit is independent of the 'think' stage because $Z/\alpha_0(N) = Z/N \to 0$ as $N \to \infty$.

It is interesting to note that since $\lambda(N)$ has a finite limit, $\bar{N}_0$ also has a finite limit. From equation (9a) it follows that

$$\lim_{N\to\infty} \bar{N}_0(N) = Z \times \lambda(\infty) \tag{12b}$$

In other words, as a system gets more and more overloaded, the vast majority of users are waiting for a response to their previous request, while only a limited number of users ($\bar{N}_0(\infty)$) are preparing their next transaction.

4. Example

This behaviour might best be described by an example. Consider the minicomputer system at the University of Nijmegen shown in **Fig. 2.** This system, running under the operating system 'UNIX', is made up of one PDP 11/45 central processing unit (1); one fixed head disc unit (5); and one movable head disc controller (2) with two movable head discs (3, 4) attached. Disc 3 contains the system information and swapping areas and is, therefore, by far the most active server. A software monitor was written which kept track of the use made of each device by all the time sharing users. At the time this study was made, only three terminals could be active at any time, so the swapping rate measured was less than that which would be expected under heavier loads. The utilisation times quoted in equation (13) below reflect the projected increase in swapping if and when more terminals are added. In particular, it was assumed that each T-S request would require a 'swap-in', and

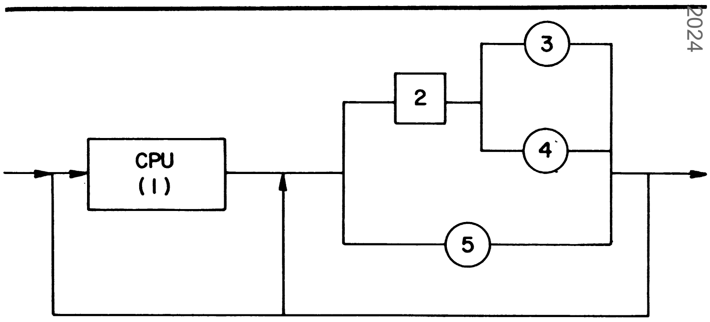

Fig. 2 Diagram of a Minicomputer System composed of one CPU (1), one disc controller (2), two movable-head discs (3, 4), and one fixed-head disc (5). Each transaction requires several I/O's, with CPU activity in between. Some I/O's (namely swaps) require several disc accesses, hence two feedback paths

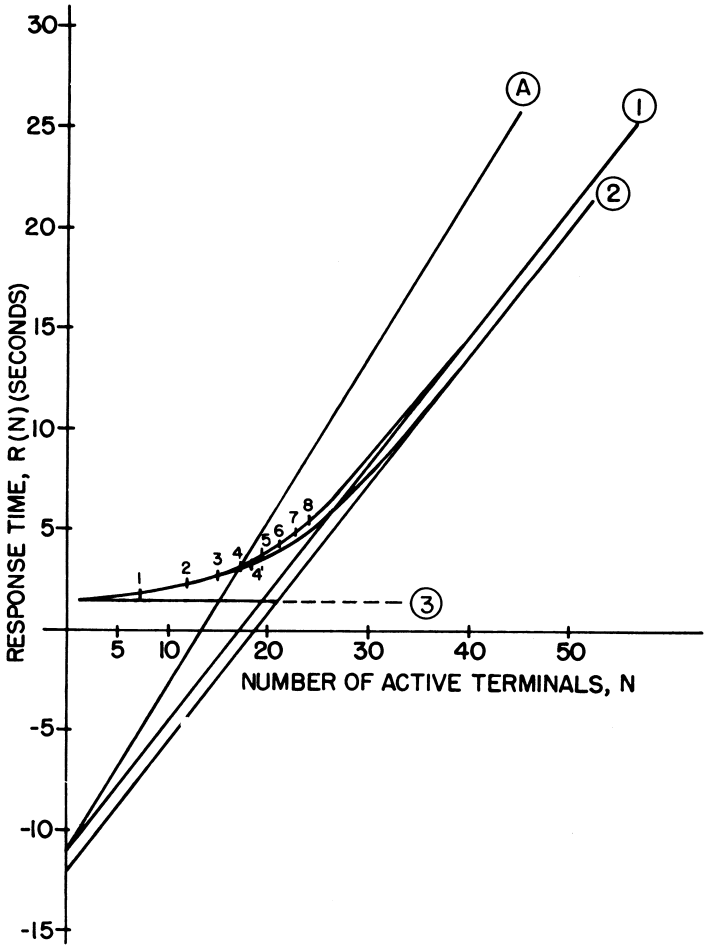

Fig. 3 Response time versus number of active terminals for two different 'think' times $Z = 11$ and $Z = 12$, for the case of unlimited storage (curves 1 and 2). Restrictive approximation 'A' is also included

0.6 'swap-outs', but no swap-outs would involve presently active transactions (no thrashing). See Section 7 and Lipsky (1977) for further details.

The expected mean times per transaction are:

$$X = \left\{X_0 = Z, \frac{0{\cdot}1244}{c}, \frac{0{\cdot}3096}{c}, \frac{0{\cdot}4545}{c}, \frac{0{\cdot}0616}{c}, \frac{0{\cdot}0499}{c}\right\} \quad (13)$$

where $\sum_{i=1}^{5} X_i = 1/c = 1/0{\cdot}70588 = 1{\cdot}4167$ seconds.

$1/c$ is the time it takes to complete one transaction if no other activities are being performed. This is also the response time with only one active terminal.

Fig. 3 shows the response time as a function of the number of users, for two 'thinking' times. We see that the slope of the asymptote (which is $1/\lambda(\infty)$) does not change with Z, nor does $R(1)$. The figure shows the obvious, that the longer the users take to think, the better is the response time, since the load on the system is less. Curve '1' (corresponding to $Z = 11$) has marked on it the mean number of transactions which are actively seeking service on the CPU or one of the I/O devices ($N - \bar{N}_0$). For example, when there are 28 users on the base system, on average four of them are 'active' and 24 are thinking. It is clear from the curve that this number grows with N, and system response time 'degrades gracefully' with increased load.

In most computers the amount of available main storage is limited, so the number of transactions which can be simultaneously active is bounded, making some of the equations described above invalid. The question then is how to incorporate into the model a restriction of the number of active users. In the next section we discuss and compare several approximate ways of doing this. The exact solutions for some simple systems are discussed by Carroll and Lipsky (1979).

5. Restrictions on number of active transactions in main memory

Before examining various ways of modelling the restriction on the number of active transactions it would be helpful to discuss the following properties. First, define $\Lambda(n)$ as the mean number of transactions which the computer can complete per second when there are *exactly* n active tasks in the subsystem. (n does not include the number 'thinking'.) It can be shown (using the normalisation of equation (13)) that:

$$\Lambda(n) = g(n-1)/g(n) \quad (14)$$

where $0 \leqslant \Lambda(n) \leqslant \Lambda(n+1)$, and the value of Λ for every n will be on or below the straight line drawn through any two successive points of $\{n, \Lambda(n)\}$ (Lipsky, 1973). Next define $W(n, \bar{n})$ to be the probability (or the fraction of time) that there are n active transactions, where

$$\sum_{n=0}^{N} W(n, \bar{n}) = 1 \quad (15a)$$

and

$$\sum_{n=1}^{N} nW(n, \bar{n}) = \bar{n}\,.$$

It follows that (Lipsky, 1973)

$$0 \leqslant \bar{\Lambda}(\bar{n}) \equiv \sum_{n=0}^{N} W(n, \bar{n})\Lambda(n) \leqslant \Lambda(\bar{n}) \quad (15b)$$

That is, the weighted average must always be less than or equal to the value of the function at the mean. The lower bound is 0, for consider the particular activity function:

$$\begin{aligned} W0(, \bar{n}) &= 1 - \frac{\bar{n}}{N} \\ W(N, \bar{n}) &= \bar{n}/N \\ W(n, \bar{n}) &= 0 \text{ otherwise} \end{aligned} \quad (16a)$$

Equations (15a) are satisfied, and

$$\bar{\Lambda}(\bar{n}) = \frac{\bar{n}}{N}\Lambda(N)\,, \quad (16b)$$

which goes to zero with increasing N, since Λ is a bounded function.

Next look at equation (8a), inserting equation (14).

$$G(N-1) = \sum_{n=0}^{N-1} g(n)\frac{Z^{N-1-n}}{(N-1-n)!} = \sum_{n=0}^{N-1} \Lambda(n+1)g(n+1) \times \frac{Z^{N-1-n}}{(N-1-n)!} \quad (17)$$

$$= \sum_{n=1}^{N} \frac{\Lambda(n)g(n)Z^{N-n}}{(N-n)!}$$

This, together with equation (9b) leads to

$$\lambda(N) = \sum_{n=1}^{N} \Lambda(n)W_N(n)\,, \quad (18)$$

where $W_N(n)$ is the same as that in equation (8b). It is seen then, that $\lambda(N)$ can be written as a weighted average over the $\Lambda(n)$'s.

Case A

From an examination of the base response curve of Fig. 3, together with equation (10) it is evident that the slope of the straight line drawn from $(0, -Z)$ to the point $(N, R(N))$ is equal to $1/\lambda(N)$. It also corresponds to having $\bar{n} = \lambda(N)R(N)$ (Little's Law) active transactions on the average. The simplest

restrictive assumption would be to suppose that when $\bar{n}$ reaches some N_m, (core saturated on average), the throughput no longer increases, but remains constant. This produces a response curve which follows the straight line for large enough N which goes through $(0, -Z)$ and crosses the response curve at N_m. Such a line for $Z = 11$ (labelled 'A') is drawn on Fig. 3 for $N_m = 4$.

The line drawn from $(0, -12)$ to its corresponding response curve at $N_m = 4$ has a slightly larger slope than that for $Z = 11$. In this approximation then, the maximum number of transactions which can be processed per unit time actually decreases with increasing Z.

Case B

A reasonable approach to restricting the number of active transactions is to introduce a 'pseudo-queue' with the parameter, t, and vary t until the mean number active ($\bar{n}$) equals the mean number allowed (N_m). The queueing model appears in **Fig. 4.** When $\bar{n}$ for $t = 0$ is greater than N_m, the waiting stage becomes active. At this point a new generating function must be defined.

$$\mathbf{G}(N, t) = \sum_{n=0}^{N} t^n G(N - n) . \qquad (19a)$$

It is not hard to show that $\mathbf{G}$ satisfies the recursive equation

$$\mathbf{G}(N + 1, t) = G(N + 1) + t\mathbf{G}(N, t) . \qquad (19b)$$

That t must be found for which:

$$\bar{F}(t, N) = N - N_m - \bar{N}_0 - \bar{N}_w = 0 \qquad (20a)$$

where $\bar{N}_0$ is defined by equation (9a), and:

$$\bar{N}_w = \sum_{n=1}^{N} nt^n G(N - n)/\mathbf{G}(N, t) . \qquad (20b)$$

This root-finding problem can be solved in various ways. It turns out that $\mathbf{G}'(N, t) = d\mathbf{G}(N, t)/dt$ satisfies the same recursive equation as $\mathbf{G}$, so it is computationally easy to apply the Newton–Raphson method for finding the appropriate t. Details are given in Appendix 1, where the auxiliary function:

$$F(t, N) = \bar{F}(t, N) \times \mathbf{G}(N, t) \qquad (21)$$
$$= (N - \bar{N}_0)\mathbf{G}(N, t) - Z\mathbf{G}(N - 1, t) - t\mathbf{G}'(N, t)$$

is used.

Curve B on **Fig. 5** shows the result of this procedure for $N_m = 4$. The curve does not follow line 'A' of Fig. 3 but in fact rises more steeply, finally following a different asymptote (see **Fig. 7**). The productivity of the system actually decreases with more users. This sort of behaviour is generally observed on overloaded systems, and is usually explained as due to 'thrashing' or increased overhead, items which are not included in this model.

This can be explained to some extent by observing that since the average number of active transactions is held constant, the fluctuations about the mean must increase with increasing N. Generally, as the number of users goes up, in order to maintain

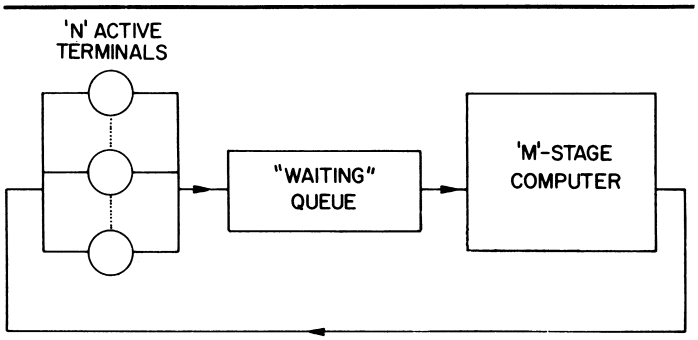

Fig. 4 Diagram of a Terminal-Computer System, with a pseudo-queue of transactions waiting to become active in the computer

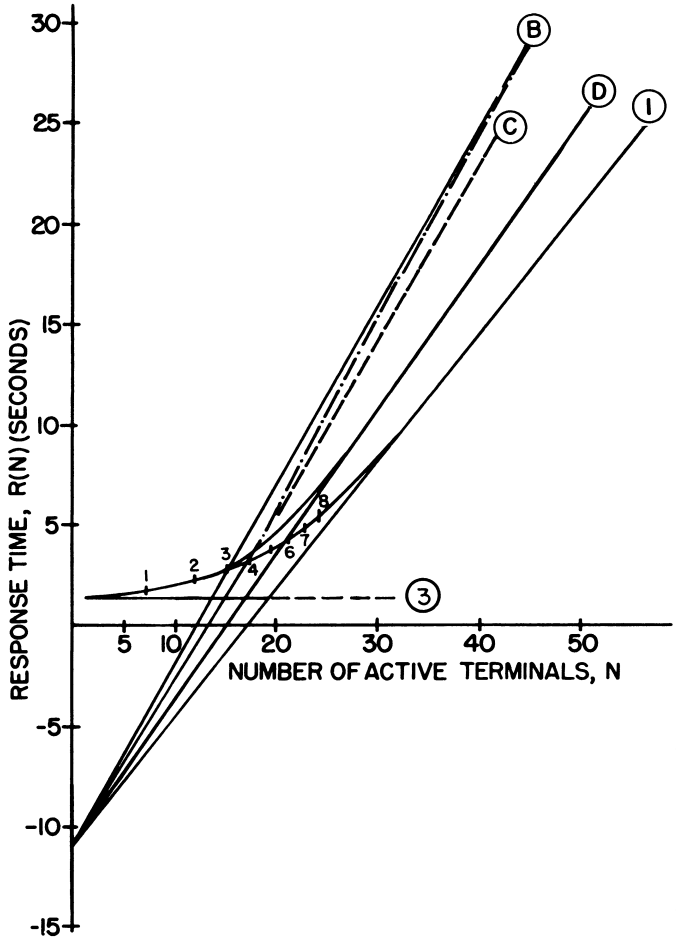

Fig. 5 Response time versus number of active terminals, and restrictive approximations B, C, and D of the text. 'Think' time is 11 seconds

a fixed average, the probability of having less than N_m tasks active must also go up, thereby giving increasing weight to the smaller values of $\Lambda(n)$ in equation (18).

Case C

Although the above results have been observed in real systems (e.g. Tiemann *et al*, 1978) it may be fortuitous to some extent, since there is no direct correlation between a transaction completing, and another jumping in to take its place. In this section it is assumed that the waiting stage is of the multiple server type in an attempt to make the 'waiting' queue respond more directly to completion of tasks. Here,

$$\mathbf{G}(N, t) = \sum_{n=0}^{N} \frac{t^n}{n!} G(N - n) \qquad (22)$$

and

$$F(t, N) = (N - \bar{n}_0)\mathbf{G}(N, t) - (Z + t)G(N - 1) \qquad (23)$$

The mathematical details are presented in Appendix 2. Discussion is similar to that in case B. These results are also plotted in Fig. 5 labelled curve **C.** Notice once again that the response time increases rapidly with increasing numbers of terminals, while the service rate actually decreases, but not quite as much as in case B.

Case D

In many computer systems, particularly where each user has a fixed partition (e.g. TSO for IBM/370 systems), the maximum number of active transactions, N_m is fixed. Then, as the load is increased, the mean number active approaches N_m. In this case

we would expect the response time to go asymptotically as

$$\frac{N}{\Lambda(N_m)} - Z .$$

The queueing theory of systems where the number of customers which can simultaneously be active in a given subsystem is restricted has not been studied in great detail. The following discussion considers an approximation which gives the correct results both when the load is small, and when the load is very large. It has been discussed by Williams and Bhandiwad (1976), Kleinrock (1976), Chandy *et al* (1975), and Carroll and Lipsky (1979).

First consider a single load dependent server whose service rate is $\Lambda(n)$, the same as that for our M-stage computer. The probability of a service completing between t and $t + \Delta t$, when there are n active tasks is

$$\Lambda(n)\Delta t = \frac{\Lambda(n)}{\Lambda(1)} \times \frac{\Delta t}{[1/\Lambda(1)]} \tag{24a}$$

In comparing with equation (3a) it is seen that

$$\alpha(n) = \Lambda(n)/\Lambda(1) \tag{24b}$$

$$t = 1/\Lambda(1) ,$$

and

$$\beta(n) = \frac{\Lambda(1)\Lambda(2) \ldots \Lambda(n)}{[\Lambda(1)]^n} \tag{24c}$$

where $\beta(0) \equiv 1$.

But, from equation (14) $\Lambda(n) = g(n-1)/g(n)$, so

$$\beta(n) = \left[\frac{g(0)}{g(1)}\right]^n \times \frac{g(0)}{g(1)} \times \frac{g(1)}{g(2)} \times \ldots \frac{g(n-1)}{g(n)} .$$

Noting that $g(0) \equiv 1$, the above leads to

$$\beta^{-1}(n) = [g(1)]^n \times g(n). \tag{24d}$$

For any load dependent server,

$$G(N) = \sum_{n=0}^{N} \frac{Z^{N-n}}{(N-n)!} \frac{t^n}{\beta(n)} = \sum_{n=0}^{N} \frac{Z^{N-n}}{(N-n)!} \left[\frac{1}{g(1)}\right]^n \times g(n)\,[g(1)]^n$$

$$= \sum_{n=0}^{N} \frac{Z^{N-n}}{(N-n)!} g(n) .$$

This is identical with equation (8a), showing that this single stage whose behaviour does not depend on Z or N, provides results which are identical to that of the M-stage computer. We must warn that this remarkable result which says that any subsystem can be replaced by a one stage 'black box', is only true for exponential servers with no restrictions on the number of active customers. Even so, it is asymptotically true (Carroll and Lipsky, 1979). See, however, Denning and Buzen (1977).

With this insight we can restrict the service rate of our equivalent server by making the following 'Natural approximations'.

$$\Lambda(n) = \Lambda(n) \qquad \text{for } n \leqslant N_m \tag{26a}$$

$$\Lambda(n) = \Lambda(N_m) \qquad \text{for } n \geqslant N_m$$

or

$$\beta(n) = [g(1)]^n/g(n) \quad \text{for } n \leqslant N_m \tag{26b}$$

$$\beta(n) = \frac{[g(1)]^{N_m}}{g(N_m)} \left[\frac{g(N_m - 1)}{g(N_m)}\right]^{n-N_m} \quad \text{for } n \geqslant N_m \tag{26c}$$

This leads to the generating function

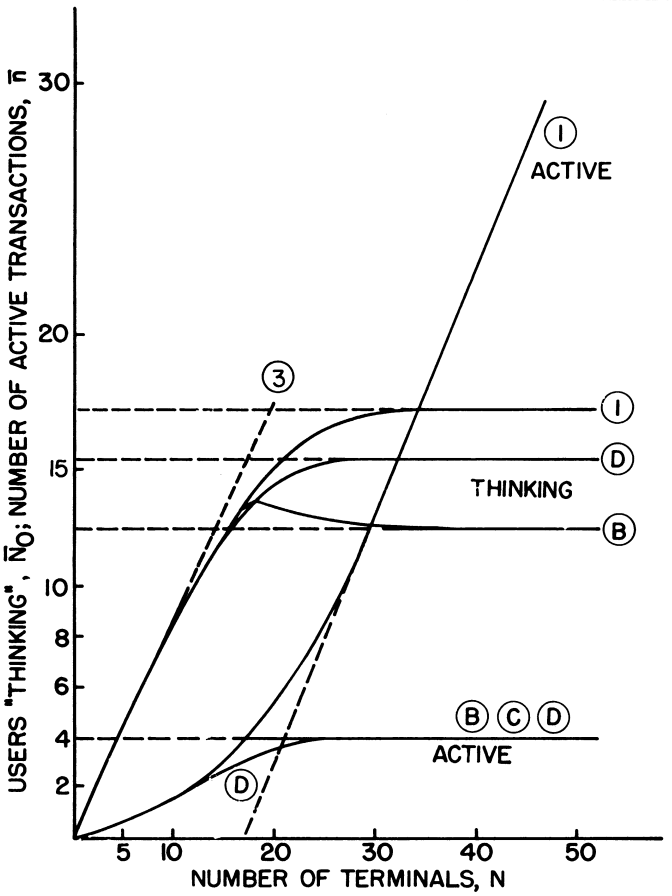

Fig. 6 Number of users thinking ($\bar{N}_0$), and number of transactions in main memory ($\bar{n}$), versus number of active terminals (N). Same systems as Figs. 3 and 5

$$G(N) = \sum_{n=0}^{N} \frac{Z^n}{n!} g(N-n) \text{ for } N \leqslant N_m \tag{27a}$$

and for $N \geqslant N_m$

$$G(N) = \sum_{n=0}^{N_m} \frac{Z^{N-n}}{(N-n)!} g(n) + g(N_m) \sum_{n=N_m+1}^{N} \frac{Z^{N-n}}{(N-n)!} \left[\frac{g(N_m)}{g(N_m - 1)}\right]^{n-N_m} \tag{27b}$$

It follows directly from equation (5b) that an arbitrary closed queueing network with exponential servers in the steady state, can be decomposed into two disjoint subsystems, where each subsystem can be replaced by an equivalent single stage. In cases where the number of active customers within each subsystem is restricted, equations (24) and (25) provide an approximation which is asymptotically correct. The excess customers are waiting to enter one or the other subsystem. This is different from the 'finite waiting room' problem (Cox, 1961), where customers are actually turned away, or the 'blocking effect' (Morse, 1958; Konheim, 1976) where one server cannot take on a new customer until another server completes its task. It is not known in general how accurate this approximation is except that it must be asymptotically correct as $N \to \infty$.

Equations 27 have been solved for our computer system with $N_m = 4$, and plotted on Fig. 5, labelled D. As expected, this curve gradually diverges from curve 1, asymptotically going to the throughput $\Lambda(N_m)$. Observe that the response time under heavy load is actually less than the response times in cases B and C even though there are fewer active transactions on the average. (B and C have an average of 4, whereas D has a maximum of 4.)

The mean number of transactions which are being simultaneously processed can be calculated by evaluating

$$\bar{n}(N) = \sum_{n=1}^{N_m} n \frac{Z^{N-n}}{(N-n)!} g(n) + N_m g(N_m) \sum_{n=N_m+1}^{N} \frac{Z^{N-n}}{(N-n)!} \left[\frac{g(N_m)}{g(N_m-1)}\right]^{n-N_m} \quad (28)$$

On **Fig. 6** are plotted $\bar{n}(N)$ together with $\bar{N}_0$, the number 'thinking', for several of the cases we have discussed. Note that $N - \bar{N}_0$ equals $\bar{n}(N)$ *plus* the number waiting to get into main memory.

6. System throughput

In the study of time sharing systems, the response time is usually the most important property to be examined. Even so, an examination of the system's throughput can provide some interesting insights into its performance characteristics. The two properties are directly related by equation (10), which is rewritten below:

$$R(N) = \frac{N}{\lambda(N)} - Z \quad (29)$$

This equation is significant, since it is universally valid, irrespective of the various service time distributions or restrictions on the computer subsystem. The throughputs of the various systems described previously are drawn on Fig. 7. Every horizontal line here, corresponds to a straight line eminating from $-Z$ on Fig. 3. Also, the line labelled 3, which corresponds to the most optimistic system behaviour:

$$\lambda(N) = N\lambda(1) \quad (30)$$

produces the constant response time $R(N) = R(1)$.

Curves B and C show clearly how system behaviour actually gets worse with increasing load. In such cases it is better to restrict the number of users than it is to allow more users to share the degraded system. Examination of Figs. 3, 5 and 7 indicates that the essential characteristics depend strongly on saturated throughput $\lambda(\infty)$, and initial throughput, $\lambda(1)$. Both of these numbers, like equation 29, are independent of various detailed assumptions of a given model, and can generally be measured or reasonably estimated, provided there are no unstable influences such as 'thrashing' (see next section) which can make response time grow exponentially (therefore making $\lambda(\infty)$ zero) rather than linearly. It is easy to show that

$$\lambda(1) = \frac{\Lambda(1)}{1 + \Lambda(1)Z} \quad (31)$$

where $\Lambda(1)$ is the c of equation 13, and is independent of Z.

Several semi-imperical approximations may suggest themselves for consideration. For instance, the response time can be assumed to follow line 3 of Fig. 3 until it crosses the appropriate asymptote (sometimes called the 'saturation point'), or it can be assumed that the response curve goes linearly through $R(1)$ and $R(2)$ (a number which must be measured or calculated independently) and continues until it crosses the appropriate asymptote. From Fig. 7, one might assume that $\lambda(N)$ is of the form $\lambda(1)[N - (N-1)^2\alpha]$, where $\alpha = 2 - \lambda(2)/\lambda(1)$ also must be evaluated independently.

These approximations can help provide 'back of the envelope' estimates of how a given system might behave for low to moderate loads, or under saturated conditions, but in the transition region (in our example, for $15 \leqslant N \leqslant 25$) much more detail is necessary. In any case, even though the system at the University of Nijmegen never had more than three users simultaneously, expansion plans were made based on the statement that 'certainly 15 terminals can be supported, but not 25', a factor of less than two in cost uncertainty.

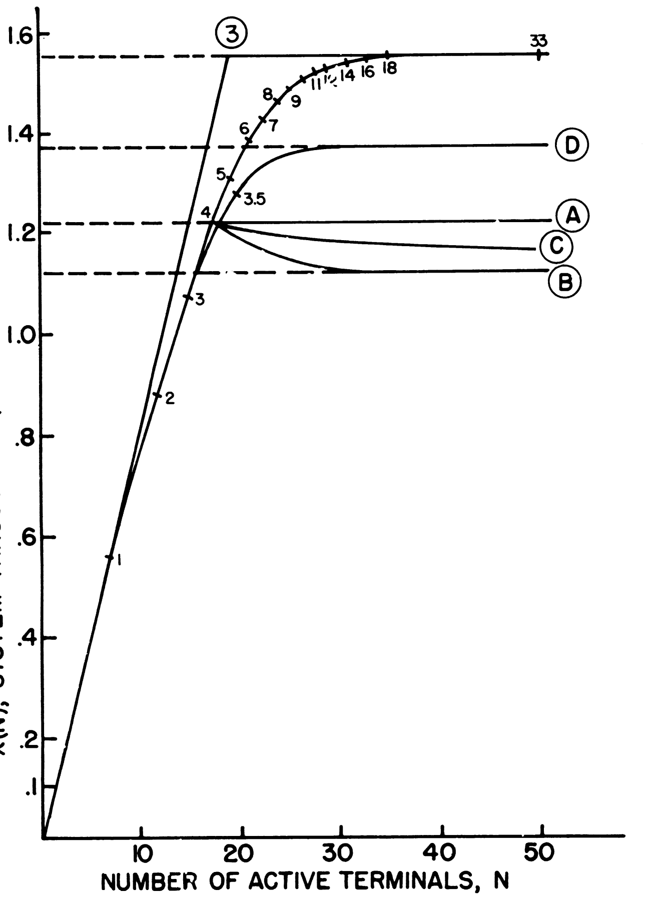

Fig. 7 Number of transactions processed per second ($\lambda(N)$) versus number of active terminals. Same systems as Figs. 3, 5 and 6

7. Comments on 'swapping' and 'thrashing'

In any time sharing system, there is usually not enough main memory to hold all material that may be needed, so most material resides on some disc file. When a user enters a command, the system program checks to see if the material needed to execute the command is already resident in memory. If it is, then the transaction becomes 'active' and enters the CPU ready queue. If, however, the material is not already resident, the system must check to see if there is space available in the memory, and where the material is stored. The action of bringing this material from disc to main memory is called a 'swap'. If there is no unused space, then the system checks to see if some of the space is taken up with material which belongs to a user who is not presently 'active'. If enough of this space can be found, then the swap proceeds, after checking to see if the old material should be saved and if so then swapping it out. If there is insufficient space available for the material, then different systems make different decisions. One common algorithm is to check to see which of the active programs has been active the longest, and if this time exceeds some preset number, say 3 seconds, then this task is swapped out, put at the end of the memory queue, and the new one is swapped in. This is done to protect against very long transactions dominating the computer.

If the load on the system becomes very great, then even short transactions will remain active for a long time. We then observe the phenomenon of a given transaction with average or less than average demand on the system, being swapped in and out more than once. This puts an even greater load on the system, thereby reducing productivity, forcing jobs to remain active longer, and thereby swap more frequently. This phenomenon is called 'thrashing'.

Although the four cases described in this paper may not cover

precisely the reality of most time sharing systems, they all have something in common. When available memory gets scarce (when the average number active approaches the number which can be held), each of the cases discussed indicates that response time goes up rapidly (degradation is not so graceful), since most transactions spend most of their time waiting for memory to become available. Thrashing cannot occur until this situation is reached, so we see that thrashing may accompany core saturation and poor response time, but it does *not* necessarily cause the poor response. It merely makes an already bad situation worse.

For the system at Nijmegen, even when there are few users, a swap occurs for one in every two transactions (new material must be brought in). As the number of users goes up, so that there are more users than there is space, this may go to one swap in for every transaction (and some fraction of a swap out). With greater increase in load, this should remain relatively constant, until the mean response time exceeds the cutoff time parameter. But from Fig. 3 and Fig. 5 that means the average transaction is already taking over 3 seconds to respond, with 5% taking more than 9 seconds. Most of the time is spent waiting for main memory to become available. An 'anti-thrashing' algorithm could be implemented which allows a program to be swapped out only after it has used a certain amount of resource time, not 'wall clock' time. The time it spends waiting for resources to become available should not be included. Such an algorithm would produce a response curve similar to those in this paper, without any explosive degradation of service with increased load.

In any case, time sharing systems can reasonably be modelled under the assumption that there is always one swap per transaction (plus a fraction more for swap outs). This will overestimate the response time when the system is under light load (and who cares?) but in the critical region, when the response curve is beginning to rise rapidly, the model can be expected to give reasonable results.

Appendix 1

We wish to find that t for a single server 'waiting' stage which allows precisely N_m to be active, on the average.
The mean number 'thinking' is

$$\bar{N}_0 = \sum_{n=0}^{N} n \frac{Z^n}{n!} \bar{G}(N-n, t)/\mathbf{G}(N, t) \tag{1.1}$$

where $\bar{G}$ is the normalising function of the entire system excluding the time sharing or 'thinking stage'. That is,

$$\bar{G}(N, t) = \sum_{n=0}^{N} t^n g(N-n)$$

and

$$\mathbf{G}(N, t) = \sum_{n=0}^{N} \frac{Z^n}{n!} \bar{G}(N-n, t) \tag{1.2}$$

From (1.1)

$$\bar{N}_0 = \sum_{n=1}^{N} \frac{Z^n}{(n-1)!} \bar{G}(N-n, t)/\mathbf{G}(N, t) = Z \times \sum_{n=0}^{N-1} \frac{Z^n}{n!} \bar{G}(N-1-n, t)/\mathbf{G}(N, t)$$
$$= Z\mathbf{G}(N-1, t)/\mathbf{G}(N, t) \tag{1.3}$$

The number in the waiting queue (using equation (4a)) is

$$N_w = \sum_{n=0}^{N} nt^n G(N-n)/\mathbf{G}(N, t) \tag{1.4}$$

Compare this with $\frac{d\mathbf{G}(N, t)}{dt}$. Since

$$\mathbf{G}(N, t) = \sum_{n=0}^{N} t^n G(N-n) \tag{1.5a}$$

we see that

$$\mathbf{G}'(N, t) = \sum_{n=1}^{N} nt^{n-1} G(N-n) = \frac{1}{t} \sum_{n=1}^{N} nt^n G(N-n), \tag{1.5b}$$

which on comparison with (1.4) leads to

$$N_w(N, t)\mathbf{G}(N, t) = t\mathbf{G}'(N, t) \tag{1.6}$$

The number in core is equal to $N - \bar{N}_0 - N_w$ and if it is desired that this number be N_m then t must be found such that

$$F(N, t) = (N - N_w - \bar{N}_0 - N_m)\mathbf{G}(N, t)$$
$$= (N - N_m)\mathbf{G}(N, t) - Z\mathbf{G}(N-1, t) - t\mathbf{G}'(N, t) \tag{1.7}$$

In order to use the Newton–Raphson method, we must also evaluate $F'(N, t)$, which is:

$$F'(N, t) = (N - N_m - 1)\mathbf{G}(N, t) - Z\mathbf{G}'(N-1, t) - t\mathbf{G}''(N, t) \tag{1.8}$$

Finally, $\mathbf{G}'$ and $\mathbf{G}''$ can be evaluated recursively from the equations obtained by differentiating the recursive forms for $\mathbf{G}(N)$. Here,

$$\mathbf{G}(N, t) = G(N) + t \times \mathbf{G}(N-1, t) \tag{1.9a}$$

Differentiating:

$$\mathbf{G}'(N, t) = t\mathbf{G}'(N-1, t) + \mathbf{G}(N-1, t) \tag{1.9b}$$

and again:

$$\mathbf{G}''(N, t) = 2\mathbf{G}'(N-1, t) + t\mathbf{G}''(N-1, t) \tag{1.9c}$$

with the initial conditions:

$$\mathbf{G}(0, t) = \mathbf{G}'(1, t) = \frac{1}{2}\mathbf{G}''(2, t) = 1 \tag{1.9d}$$
$$\mathbf{G}'(0, t) = \mathbf{G}''(0, t) = \mathbf{G}''(1, t) = 0$$

All that remains is to iteratively evaluate $t \leftarrow t - F/F'$ until some predetermined accuracy is satisfied.

Appendix 2

We now wish to find that t for multiple server 'waiting' stage which allows precisely N_m to be active on the average. As in Appendix 1,

$$\bar{N}_0 = Z\mathbf{G}(N-1, t)/\mathbf{G}(N, t)\,. \tag{2.1}$$

But now,

$$N_w = \sum_{n=0}^{N} n \frac{t^n}{n!} G(N-n)/\mathbf{G}(N, t) = t\mathbf{G}(N-1, t)/\mathbf{G}(N, t) \tag{2.2}$$

This yields

$$F(N, t) = (N - N_w - \bar{N}_0 - N_m)\mathbf{G}(N, t)$$
$$= (N - N_m)\mathbf{G}(N, t) - (Z + t)\mathbf{G}(N-1, t)\,. \tag{2.3}$$

Also

$$F'(N, t) = (N - N_m)\mathbf{G}'(N, t) - (Z + t)\mathbf{G}'(N-1, t) - \mathbf{G}(N-1, t) \tag{2.4}$$

and

$$\mathbf{G}'(N, t) = t\mathbf{G}(N-1, t)/\mathbf{G}(N, t) \tag{2.5}$$

Acknowledgements
The author would like to thank Dr Jan van den Bos and members of the Computer Graphics Group at the University of Nijmegen (particularly Sjors Rolf and Hendrick-jan

Thomassen) for their close co-operation and hospitality during his stay there. Thanks are also due to Geoffrey Beaumont, Alan Davies and Anthony Unwin (now of University College Dublin) for stimulating discussions while visiting Royal Holloway College (London) with partial support from the SRC.

References

BHANDIWAD, R. A. and WILLIAMS, A. C. (1974). Queueing network models of computer systems, in *Proc. Third Annual Symp. Computer Systems*, Univ. Texas.

BOYSE, J. W. and WARN, D. R. (1975). A straight-forward model for computer performance prediction, *Computing Surveys*, Vol. 7 No. 2, pp. 73-93.

BRANDWAJN, A. (1974). A model of a time-sharing system solved using equivalence and decomposition methods, *Acta Inf*, Vol. 4 No. 1, pp. 11-47.

BUZEN, J. P. (1973). Computational algorithms for closed queueing networks with exponential servers, *CACM*, Vol. 16 No. 9, pp. 527-539.

CARROLL, J. L. and LIPSKY, L. (1979). Closed queueing networks with population size constraints, including a recursive algorithm for M/G/1/N systems, unpublished.

CHANDY, K. M., HERZOG, V. and WOO, L. (1975). Parametric Analysis of Queueing Networks, *IBM J. Res. Devel*, Vol. 19, pp. 36-42.

CHIU, W., DUMONT, D. and WOODS, R. (1975). Performance analysis of a multi-programmed computer system, *IBM J. R & D*, Vol. 19, pp. 263-271.

COX, D. R. and SMITH, W. L. (1961). *Queues*, Methuen (London) and John Wiley and Sons (New York).

DENNING, P. J. and BUZEN, J. P. (1978). The operational analysis of queueing network models, *Computing Surveys*, Vol. 10 No. 3, pp. 225-261.

GIAMMO, T. (1976). Validation of a computer performance model of the exponential queueing network family, *Acta Informatica*, Vol. 7, pp. 137-152.

GORDON, W. J. and NEWELL, G. F. (1967). Closed queueing systems with exponential servers, *Operations Research*, Vol. 15, pp. 254-265.

JACKSON, J. R. (1963). Jobshop-like queueing systems, *Management Science*, Vol. 10, pp. 131-142.

KLEINROCK, L. (1975). *Queueing systems*, Vol. 1, Wiley-Interscience, NY.

KLEINROCK, L. (1976). *Queueing systems*, Vol. 2, Wiley-Interscience, NY.

KONHEIM, A. G. and REISER, MARTIN (1976). A Queueing Model with Finite Waiting Room and Blocking, *JACM*, Vol. 23, p. 328.

LIPSKY, L. (1973). A simple analysis of a multiprogramming computer system, Lincoln Computing Facility (LCF) Report, University of Nebraska, Lincoln, Nebraska 68588.

LIPSKY, L. (1974). Device utilization in a finite, but otherwise arbitrary closed queueing network with exponential servers, LCF Report, University of Nebraska, Lincoln.

LIPSKY, L. (1974). Use of queueing theory for modelling a computer system, in *Proc. SHARE*, Vol. XLIII, p. 602.

LIPSKY, L., OZORIO, A., BEGLEY, A. and COLEMAN, R. (1977). Comparison of Performance Measurements of a Computer System with a Queueing Theory Model, *Int. Comp. Symp.*, North-Holland, pp. 461-466.

LIPSKY, L. (1977). Modelling of the PDP11/45 Mini-Computer System Operated by the Computer Graphics Group at Nijmegen University, Computer Graphics Group, Informatica, Katholieke Universiteit, Nijmegen, The Netherlands.

LIPSKY, L. and CHURCH, J. D. (1977). Applications of a Queueing Network Model for a Computer System, *Computing Surveys*, Vol. 9 No. 3, pp. 205-221.

MORSE, P. M. (1958). *Queues, Inventories and Maintenance*, John Wiley and Sons.

SCHERR, A. L. (1967). *An analysis of time shared computer systems*, MIT Press, Cambridge, Mass.

TIEMANN, D., NESSER, R. and YEH, T. N. (1978). A Simple Queueing Theory Model of NUROS, LCF Report, University of Nebraska, Lincoln.

WILLIAMS, A. and BHANDIWAD, R. A. (1976). A Generating Function Approach to Queueing Network Analysis of Multiprogrammed Computers, *Networks*, Vol. 6, pp. 1-22, John Wiley & Sons.

Calls for papers

The First British National Conference On Databases will be held on July 13-14, 1981 at Jesus College, Cambridge, following the success of the International Conference On Databases (ICOD-1) at Aberdeen. The BNCOD series is meant to focus primarily on British work, although overseas papers are also welcome. This conference is being organised jointly by the Aberdeen University Computing Science Department, The British Computer Society and Middlesex Polytechnic. *It is intended to publish the conference proceedings.*

Research papers are invited on *all aspects* of data bases such as: data modelling; data base design; restructure and reorganisation; performance optimisation; data dictionaries and design tools; privacy, integrity, consistency and recovery; end user facilities; data base machines; distributed data bases.

Implementation oriented research work is particularly welcome.

Please submit an abstract of about 500 words by **2 February 1981** and five copies of the complete paper (not more than 8,000 words each) by 2 April 1981 at the latest. The abstracts will be used for a preliminary selection, the final selection based on complete papers being made by the beginning of May 1981. All communications should be addressed to

Dr S. M. Deen (conference chairman)
Department of Computing Science
University of Aberdeen
Aberdeen AB9 2UB
Telephone 0224-40241 ext 6421 or 6417 Telex 73458

Papers are invited for the international symposium on algorithmic languages to be held in Amsterdam, The Netherlands on 26-29 October 1981. Authors are invited to send five copies of a full draft, not exceeding 5000 words, by **1 February 1981,** to the program secretary: J. C. van Vliet, Mathematical Centre, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands. Further details may be obtained from him also.

The IEEE Computer Society conference on Pattern Recognition and Image Processing will be held in conjunction with ACM SIGGRAPH 81 at the Hyatt Regency Hotel, Dallas, Texas on 3-5 August 1981. A joint session on topics of common interest to image processing and computer graphics will be held. Papers are invited on all aspects of the subject. Four copies of the complete draft papers should be submitted by **15 January 1981** to Azriel Rosenfeld, Computer Science Center, University of Maryland, College Park, MD 20742, USA.

The *Australian Computer Journal* will publish a special issue on 'Recent developments in computer networks', with special attention to developments in Australia, in May 1981. Both full papers and short communications will be considered. Prospective authors should write to Dr John Lions, Guest Editor, ACJ Special issue on Computer Networks, Department of Computer Science, University of New South Wales, Kensington, NSW 2033, Australia.