THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 54, No. 3, March 1975
Printed in U.S.A.

Analysis and Optimal Design of a Multiserver, Multiqueue System With Finite Waiting Space in Each Queue

By B. A. WHITAKER

(Manuscript received July 29, 1974)

An analysis of a particular type of multiserver, multiqueue system is presented in which each queue has a finite number of waiting positions and the waiting positions are not *vacated until service is completed. Thus, several customers in one queue can be served simultaneously. The steady-state distribution of states is derived and is used to obtain the probability of loss for each queue and the average delay of the system. This analysis is then used in the development of a design procedure to determine the minimum-cost configuration of waiting positions and servers to meet specified single-hour grade-of-service constraints. The results are applicable to the design of systems that utilize automatic call distributors. While this model does not include such effects as day-to-day variation and noncoincidence of peak loads among trunk groups, nevertheless the results properly reflect for the first time the interactions among the trunk groups terminated on the automatic call distributor and the attendants at the automatic call distributor.*

I. INTRODUCTION

The purpose of this paper is to present an analysis of a particular type of multiserver, multiqueue system in which each queue has a finite number of waiting positions and the waiting positions are *not* vacated until service is completed. In particular, the steady-state distribution of states are derived, and expressions for the probability of loss for each queue and the average delay are given. It is shown that the queuing model described is representative of systems characterized by a finite number of trunk groups that carry calls to a group of attendants who then perform some service for the caller. (One such system is used in the directory assistance service provided by the telephone company.) Results are given to illustrate the effects of varying the number of servers and number of positions in each queue. Finally, a design procedure to determine the minimum-cost configura-

tions for such systems under various grade-of-service constraints is developed. This procedure ignores such effects as day-to-day variation, noncoincidence of peak loads among incoming trunk groups, and retrials of blocked calls, which should be investigated in the development of procedures for traffic engineering and administration. However, as in most cases, it is difficult, if not impossible, to obtain analytical results with these effects included. This paper should provide useful insight that can later be incorporated in a complete traffic engineering procedure.

The system analyzed consists of l input queues each with a finite number of waiting positions, N_i ($i = 1, \cdots, l$), which have full access to M servers. When an arrival seizes one of the M servers, it does *not* vacate a waiting position, but remains in the position until its service has been completed. This characteristic, which allows calls that are not at the head of a queue to be in service, distinguishes this system from the usual queuing system. In particular, the system is no longer completely described by the number of calls in each queue since a record of the number of calls from each queue that are in service must be kept. An arrival that finds all the waiting positions for its queue occupied is cleared or lost from the system. An arrival finding no idle servers but at least one vacant waiting position in its queue enters the queue and is delayed until its service begins. In the context of directory assistance systems, the input queues are the trunk groups and the waiting positions are represented by the trunks. The information operators are the servers.

In telephone traffic theory, the described system has been referred to as a combined loss-and-delay system. Previous work in this subject can be segmented into three parts:

(*i*) One input flow—one queue.[1–8]
(*ii*) Several input flows—one queue.[9–11]
(*iii*) Several input flows—several queues.[12–14]

The last segment, of which this analysis is a part, has been investigated by Kühn. He analyzed systems with $g > 1$ queues, each with a finite number of waiting positions s_i ($i = 1, 2, \cdots, g$). Associated with each queue is a Poisson arrival process with mean rate λ_i, which is assumed to be independent of the others. An arrival that finds all waiting positions in its queue occupied is lost. Arrivals that are not cleared from the system are served by one of n servers. The service time distribution for the ith server is exponential with a mean rate ϵ_i. When a server becomes idle, queue i is chosen to receive service with probability p_i. Within a queue, calls can be selected randomly, first-come, first-served, or according to a priority scheme.

As Kühn indicates, analytical solutions to these systems exist only for special cases. In general, the linear equations representing the equilibrium conditions must be solved numerically for the particular values of the parameters. However, Kühn gives a solution to one particular system, which will now be discussed. In this system, it is assumed that the service rate for all servers is identical ($\epsilon_i = \epsilon$) and the interqueue discipline is defined by the p_j's that are

$$p_j = \frac{Z_j}{\sum_{k=1}^{g} Z_k} \qquad (j = 1, \cdots, g),$$

where Z_j is the number of waiting positions occupied in the jth queue.

The system analyzed in this paper is an extension of the one examined by Kühn, since an arrival does not release a waiting position until his service has been completed. This complicates the state analysis, since it is now not sufficient to know the number of servers that are busy to determine the equilibrium equations; information as to the number of calls from each queue that are in service must be included.

Kühn indicates also that, for the above interqueue discipline, the waiting time distribution can be found numerically only for small systems. The calculation of the waiting time is complicated by the fact that an arrival's waiting time is influenced by the number of arrivals that occur *after* it has entered the system. More is said about this difficulty later.

II. MATHEMATICAL FORMULATION

In this section, a mathematical model of the queuing system is formulated. Equilibrium equations are given and their solutions derived.

2.1 Queuing model

The queuing system consists of l input queues, each with a finite length denoted by N_i, $i = 1, 2, \cdots, l$. Requests for service arrive at queue i according to a Poisson distribution with mean rate, λ_i. If we let $A_i(t)$ denote the number of arrivals at queue i in $(0, t)$, then

$$P[A_i(t) = k] = \frac{(\lambda_i t)^k}{k!} e^{-\lambda_i t} \qquad (k = 0, 1, 2, \cdots). \tag{1}$$

The arrival process at queue i is assumed to be independent of the arrival process at each of the other queues. Arrivals from each queue have full access to a group of M servers. The service time distributions

of the servers, denoted by $H(t)$, are independent and identical exponential distributions with mean service rate μ, i.e.,

$$H(t) = \begin{cases} 1 - e^{-\mu t} & (t \geqq 0) \\ 0 & (t < 0). \end{cases} \tag{2}$$

Since the arrival process is Poisson and the service process is exponential, the queuing model is a multidimensional birth-death process.

Arrivals to queue i that find N_i waiting positions in queue i occupied are not allowed to enter the system. If an arrival to queue i finds at least one unoccupied position in queue i but all M servers busy, it enters the queue and waits as long as necessary for service. Within queue i, arrivals enter service on a first-in, first-out (FIFO) basis. An arrival that finds at least one unoccupied position in its queue and at least one free server immediately enters service. When an arrival enters service, it does not release a waiting position but remains in the queue until its service has been completed. Hence, the word "queue" is being used in a nonstandard manner and refers to the number of calls waiting for service *and* in service. As discussed earlier, an example of such a queue is a group of trunks that carry calls into a switchboard.

The interqueue service discipline—the order in which the queues receive service—is characterized by the number of calls waiting for service. When a server becomes free, queue i receives service with a probability, p_i, which is the proportion of queue-i calls waiting for service. If we denote the number of calls in queue i by n_i and the number of calls in queue i that are in service by m_i, then this probability, which is dependent on $(n_1, n_2, \cdots, n_l, m_1, \cdots, m_l) \equiv (\mathbf{n}, \mathbf{m})$, can be expressed as

$$p_i(\mathbf{n}, \mathbf{m}) = \frac{n_i - m_i}{\sum_{j=1}^{l} (n_j - m_j)} = \frac{n_i - m_i}{\sum_{j=1}^{l} n_j - M} \qquad \left(\sum_{j=1}^{l} n_j \geqq M + 1 \right)$$
$$(i = 1, 2, \cdots, l). \tag{3}$$

The effects of other interqueue service disciplines have been investigated, but will not be discussed here.

The fact that arrivals remain in the queue during service distinguishes this queuing system from the standard system, since the amount of information required to fully describe a state of the system is increased. The system is also complicated by the fact that the interqueue service discipline is state-dependent. However, as is shown in later sections, this "complication" leads to a closed-form solution of the

equilibrium equations, which is generally not the case for such systems. The equations of equilibrium are given in the following section.

2.2 Equilibrium equations

The system described in the previous section is characterized by a finite number of states that indicate the number of calls in each queue and, of these calls, the number receiving service. We denote by $(n_1, n_2, \cdots, n_l, m_1, \cdots, m_l)$ the state in which there are n_i calls in queue i with m_i of these calls in service. For notational simplification, we also refer to the state in vector notation as $(\mathbf{n}, \mathbf{m})$. In this notation, $(\mathbf{n}_{i+}, \mathbf{m})$ represents the state $(n_1, n_2, \cdots, n_i + 1, \cdots, n_l, m_1, \cdots, m_l)$ and similarly $(\mathbf{n}_{i-}, \mathbf{m}) \equiv (n_i, \cdots, n_i - 1, \cdots, m_l)$. It should be clear that, if the total number of calls in the system is less than or equal to M, then $n_i = m_i$ for all i.

Assuming stationarity, let $P(\mathbf{n}, \mathbf{m})$ be the probability that at an arbitrary instant of time the system is in state $(\mathbf{n}, \mathbf{m})$. Moreover, if the arrival processes to the system are Poisson, the equilibrium-state distribution $\{P(\mathbf{n}, \mathbf{m})\}$ at an arbitrary instant is equal to the equilibrium-state distribution at the instant of an arrival. By equating the rate into a state to the rate out of a state, we can write the equilibrium-state equations where we have introduced the function

$$u(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leqq 0 \end{cases}$$

to include the boundary conditions and $a_i = \lambda_i/u$:

$$\left\{\sum_{i=1}^{l} [a_i u(N_i - n_i) + n_i]\right\} P(\mathbf{n}, \mathbf{m}) = \sum_{i=1}^{l} a_i u(n_i) P(\mathbf{n}_{i-}, \mathbf{m}_{i-}) + \sum_{i=1}^{l} (n_i + 1) u(N_i - n_i) P(\mathbf{n}_{i+}, \mathbf{m}_{i+})$$
$$\left(\sum_{i=1}^{l} n_i < M\right) \qquad (0 \leqq n_i \leqq N_i) \qquad i = 1, 2, \cdots, l \tag{4}$$

$$\left\{\sum_{i=1}^{l} [a_i u(N_i - n_i) + m_i]\right\} P(\mathbf{n}, \mathbf{m}) = \sum_{i=1}^{l} a_i u(n_i) P(\mathbf{n}_{i-}, \mathbf{m}_{i-}) + \sum_{i=1}^{l} m_i u(N_i - n_i) P(\mathbf{n}_{i+}, \mathbf{m})$$
$$+ \sum_{i=1}^{l} \sum_{\substack{j=1 \\ j \neq i}}^{l} (m_i + 1) u(N_i - n_i) u(m_j) P(\mathbf{n}_{i+}, \mathbf{m}_{i+,j-})$$
$$\left(\sum_{i=1}^{l} n_i = M\right) \qquad (0 \leqq n_i \leqq N_i) \qquad i = 1, 2, \cdots, l \tag{5}$$

$$\left\{\sum_{i=1}^{l}\left[a_i u(N_i - n_i) + m_i\right]\right\} P(\mathbf{n}, \mathbf{m}) = \sum_{i=1}^{l} a_i u(n_i) P(\mathbf{n}_{i-}, \mathbf{m})$$
$$+ \sum_{i=1}^{l} \frac{m_i(n_i + 1 - m_i)}{\left(\sum_{k=1}^{l} n_k + 1 - M\right)} u(N_i - n_i) P(\mathbf{n}_{i+}, \mathbf{m})$$
$$+ \sum_{i=1}^{l} \sum_{\substack{j=1 \\ j \neq i}}^{l} \frac{(m_i + 1)(n_j + 1 - m_j)}{\left(\sum_{k=1}^{l} n_k + 1 - M\right)} u(N_i - n_i) u(m_j) P(\mathbf{n}_{i+}, \mathbf{m}_{i+,j-})$$
$$\left(\sum_{i=1}^{l} n_i > M\right) \qquad (0 \leqq n_i \leqq N_i) \qquad i = 1, 2, \cdots, l. \quad (6)$$

Equations (4) to (6), together with the normalization condition

$$\sum_{n_1=0}^{N_1} \cdots \sum_{n_l=0}^{N_l} \sum_{m_1=0}^{\min(M,N_1)} \cdots \sum_{m_l=0}^{\min(M,N_l)} P(\mathbf{n}, \mathbf{m}) = 1, \quad (7)$$

where all nonexistent states, such as states in which both $n_i = 0$ and $m_i > 0$, in the sum are assumed to have probability zero, determine the equilibrium-state distribution.

2.3 Steady-state solution

Since the process described by the equilibrium equations is a finite-state birth-death process in which the arrival rate *into* the system is always less than the service rate of the system as a result of overflow from the finite queue, a unique solution to eqs. (4) to (7) exists, and the solution is a genuine probability distribution.[15] This solution is given in terms of $P(\mathbf{0}, \mathbf{0})$ by

$$P(\mathbf{n}, \mathbf{m}) = \begin{cases} \prod_{i=1}^{l} \frac{a_i^{n_i}}{n_i!} P(\mathbf{0}, \mathbf{0}) & \left(\sum_{i=1}^{l} n_i \leqq M\right) \quad (8) \\ \dfrac{\begin{bmatrix} \sum_{i=1}^{l} n_i - M \\ n_1 - m_1, \cdots, n_l - m_l \end{bmatrix} \begin{bmatrix} M \\ m_1, m_2, \cdots, m_l \end{bmatrix} \prod_{i=1}^{l} a_i^{n_i}}{M!\, M^{\sum n_i - M}} P(\mathbf{0}, \mathbf{0}) & \left(\sum_{i=1}^{l} n_i > M\right), \quad (9) \end{cases}$$

where $P(\mathbf{0}, \mathbf{0})$ is determined from (7). The general solution was determined from examination of various small systems. By substitution, it can be shown that this solution in fact satisfies the equilibrium-state equations (4) to (6).

When the number of calls in the system is less than or equal to the number of servers, no one is waiting for service and the queues have no interaction. This fact is shown in (8) by the product form of the solu-

tion. However, as the number of calls increases to the point where calls are waiting in more than one queue, the queues no longer are acting independently.

Since the queues behave independently as long as there are free servers, we would expect that, as the number of servers was increased, the system would approach l independent loss systems. By examining the marginal probabilities, it can be shown that this is, in fact, true when $M \geqq \sum_{i=1}^{l} N_i$. That is, the marginal state probability of n_1 calls in queue 1 is

$$P(n_1) = \sum_{n_2} \cdots \sum_{n_l} P(\mathbf{n}, \mathbf{m}) = \frac{a_1^{n_1}/n_1!}{\sum_{k=1}^{N_1} a_1^k/k!}, \tag{10}$$

which is identical to the state probability for a pure loss system.

If no calls were blocked from the system, then the system would act as a pure delay system with the offered load $a = \sum_{i=1}^{l} a_i$. This is easily shown by taking the limit of (8) and (9) as $N_i \to \infty$ for all $i = 1, 2, \cdots, l$.

We first consider the case in which $\sum_{i=1}^{l} n_i \leqq M$. Since the number of calls in each queue is unrestricted, it is easily seen that the multinominal expansion of $(a_1 + \cdots + a_l)^{\Sigma n_i}$ divided by $(\sum_{i=1}^{l} n_i)!$ can be obtained from (8). That is,

$$\lim_{\substack{N_i \to \infty \\ i=1,2,\cdots,l}} \sum_{\Sigma n_i = k} P(\mathbf{n}, \mathbf{m}) = \frac{(a_1 + \cdots + a_l)^k}{k!} P(\mathbf{0}, \mathbf{0}) \qquad \left(\sum_{i=1}^{l} n_i < M \right).$$

Hence,

$$P\left(\sum_{i=1}^{l} n_i = k \right) = \frac{a^k}{k!} P(\mathbf{0}, \mathbf{0}) \qquad (k < M). \tag{11}$$

For the case in which $\sum_{i=1}^{l} n_i \geqq M$, first note that, by the Vandermonde convolution of multinomial coefficients,

$$\sum_{m_1} \cdots \sum_{m_l} \begin{bmatrix} \sum_{i=1}^{l} n_i - M \\ n_1 - m_1, \cdots, n_l - m_l \end{bmatrix} \begin{bmatrix} M \\ m_1, \cdots, m_l \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{l} n_i \\ n_1, n_2, \cdots, n_l \end{bmatrix}. \tag{12}$$

Therefore,

$$P(\mathbf{n}) = \sum_{m} P(\mathbf{n}, \mathbf{m}) = \frac{\left(\sum_{i=1}^{l} n_i \right)!}{M!\, M^{\Sigma n_i - M}} \prod_{i=1}^{l} \frac{a_i^{n_i}}{n_i!} P(\mathbf{0}, \mathbf{0}) \qquad \left(\sum_{i=1}^{l} n_i \geqq M \right), \tag{13}$$

and thus we can denote this state probability by

$$P\left(\sum_{i=1}^{l} n_i = k\right) = \sum_{l} \cdots \sum_{\substack{\sum n_i = k}} \frac{k!}{M!\,M^{k-M}} \prod_{i=1}^{l} \frac{a_i^{n_i}}{n_i!}\, P(\mathbf{0}, \mathbf{0})$$

$$(k \geqq M). \quad (14)$$

Consequently,

$$\lim_{\substack{N_i \to \infty \\ i=1,\cdots,l}} P\left(\sum_{i=1}^{l} n_i = k\right) = \frac{a^k}{M!\,M^{k-M}}\, P(\mathbf{0}, \mathbf{0}) \qquad (k \geqq M). \quad (15)$$

The normalization constant is then expressed as

$$P(\mathbf{0}) = \left[\sum_{k=0}^{M-1} \frac{a^k}{k!} + \sum_{k=M}^{\infty} \frac{a^k}{M!\,M^{k-M}}\right]^{-1} \qquad (0 \leqq a < M). \quad (16)$$

Hence, comparison of (11), (15), and (16) with the pure delay system completes the proof.

2.4 *Blocking and delay probabilities*

In the analysis and design of queuing systems, performance measures for each configuration must be calculated. In telephone traffic theory, these performance measures are generally referred to as "grades of service." Two such measures of the grade of service are:

(*i*) For loss systems, the blocking probability or probability of loss.
(*ii*) For delay systems, the average delay experienced by calls that enter the system. The average delay $\bar{W}(s, a)$ for a pure delay system is expressed in terms of the Erlang delay formula as

$$\bar{W}(s, a) = \frac{C(s, a)}{(s - a)\mu}. \quad (17)$$

In the system described in Section 2.1, the blocking probabilities for each input queue, the average delay experienced by calls that enter the system, and the average delay of only those customers who experience a positive delay are important characteristics to be examined. The latter is not used in the remaining analysis.

The blocking probability for queue i is defined as the probability that an arrival to queue i finds N_i calls in the queue. This probability, which is denoted by $B_i(\mathbf{N}, M, \mathbf{a})$, is a function not only of the number of positions in queue i and the offered load to the queue, but also of the traffic load offered to each of the other queues, the number of positions in each of these queues, and the number of servers. Recalling that, for systems with Poisson input, the state probabilities at an arbitrary instant are equal to the state probabilities at arrival times,

$B_i(\mathbf{N}, M, \mathbf{a})$ is calculated from the marginal distribution for queue i as

$$B_i(\mathbf{N}, M, \mathbf{a}) = \sum_{n_1} \cdots \sum_{n_{i-1}} \sum_{n_{i+1}} \cdots \sum_{m_1} \cdots \sum_{m_l} P(n_1, \cdots, N_i, \cdots, n_l, \mathbf{m}). \quad (18)$$

The average delay experienced by calls that enter the system (successfully occupy a waiting position in their queue) is denoted by $\bar{D}(\mathbf{N}, M, \mathbf{a})$. The delay calculated for this system is the overall mean waiting time measured from an arrival's entry into its queue until its service begins. Hence, it does not include service time of the call. It should also be noted that arrivals into the system that find at least one free server experience no delay.

Since the average number of calls waiting for service must be finite and the mean waiting time is finite as a result of the loss structure of the system, the well-known equation of Little,[16] $L = \lambda W$, can be used to calculate $\bar{D}(\mathbf{N}, M, \mathbf{a})$. In particular, we must define our "queue length" as the total number of calls waiting for service and "λ" is defined as the effective arrival rate *into* the system. Hence,

$$\bar{D}(\mathbf{N}, M, \mathbf{a}) = \frac{\sum_{l} \cdots \sum_{\substack{\sum_{i=1}^{l} n_i > M}} \sum_{m_1} \cdots \sum_{m_l} \left[\left(\sum_{i=1}^{l} n_i - M \right) P(\mathbf{n}, \mathbf{m}) \right]}{\sum_{i=1}^{l} \lambda_i [1 - B_i(\mathbf{N}, M, \mathbf{a})]}. \quad (19)$$

Calls that are blocked from the system do not enter the queue and hence do not affect the average queue length. Consequently, they are not included in the arrival rate *into* the system. The numerator of (19) is the average number of waiting calls. It should be apparent that the average delay for any particular queue can easily be obtained by using the appropriate marginal probabilities. Also, the conditional average delay, the average delay experienced by only those that must wait, is found by dividing (19) by the probability of being delayed.

For some design purposes, it might be deemed necessary to constrain the probability of waiting longer than some time, t_0, to be less than a specified value. In this case, the waiting time distribution for each queue must be obtained. For the interqueue discipline examined for this system, the calculation of the waiting-time distribution is extremely difficult. (Kühn[13] mentions that, for his problem, numerical techniques can be used for very small systems, after which approximate methods must be formulated.) The difficulty in determining the waiting-time distribution lies in the fact that the time a particular call must wait for service is not just a function of the number of calls in

the system when it arrives, as is usually the case. In particular, the waiting time of a call is related to the number of calls that arrive after the particular call enters the system, since queues are chosen for service according to their queue lengths at a service completion, so that later calls can "pass" earlier ones. For this reason, the waiting time distributions are not calculated in this study.

2.5 Macrostate analysis

As discussed in the previous section, blocking probabilities and average delay values are of interest to system designers. Using the state probabilities given in (8) and (9), it is possible to obtain not only the overall average delay, $\bar{D}$, as shown in (19), but also the average delay, $\bar{D}_i$, for queue i. In certain types of design, we might want to engineer the system so that the average delay in *every* queue is less than a specified level. In such cases, $\bar{D}_i$ would be needed. However, for this study, we consider only the overall average delay.

Therefore, it is apparent from (19) that, for computational purposes, we only need to know the number of calls in each queue without distinguishing between those in service and those waiting. If we denote by $\mathbf{n}$ the state $(n_1, n_2, \cdots, n_l)$, we can find the steady-state probabilities $P(\mathbf{n})$ from (8) and (9). Of course, we could have written the state equations directly and solved this easier set of equations.[17] However, for further studies, it is essential to know the probabilities $P(\mathbf{n}, \mathbf{m})$.

Since the state probabilities for $(\mathbf{n}, \mathbf{m})$ in which $\sum_{i=1}^{l} n_i \leqq M$ are independent of $\mathbf{m}$, $P(\mathbf{n}) = P(\mathbf{n}, \mathbf{m})$. To obtain $P(\mathbf{n})$ for $\sum_{i=1}^{l} n_i > M$, we sum $P(\mathbf{n}, \mathbf{m})$ given by (9) over all possible values of $\mathbf{m}$. Using the Vandermonde multinomial convolution, we find that

$$P(\mathbf{n}) = \sum_{m_1} \cdots \sum_{m_l} P(\mathbf{n}, \mathbf{m}) = \frac{\left(\sum_{i=1}^{l} n_i\right)!}{M!\, M^{\sum n_i - M}} \prod_{i=1}^{l} \frac{a_i^{n_i}}{n_i!} P(\mathbf{0}), \qquad (20)$$

where $P(\mathbf{0})$ is the normalization constant. We can calculate the blocking probabilities and the average delay as before. The number of states is $\prod_{i=1}^{l} (N_i + 1)$.

It should be repeated that the macrostate probabilities are of use *only* if one is interested in the overall mean delay. To calculate the individual average delays, one must use the microstate probabilities.

III. RESULTS

In this section, we investigate the effects of varying N_i and M on the blocking probabilities for each queue and the overall average delay

of a call. In particular, these effects are illustrated by comparing the results obtained from the analysis presented in this paper, for a particular example, with the results obtained if independence is assumed. This "independence assumption" is often used in practice to determine the number of positions for each queue and the number of servers needed. In essence, this assumption permits a designer to design each queue (trunk group) independently of the other queues and the number of servers, and the number of servers is determined assuming that no calls are blocked in the queues. It is shown that this assumption is generally not even a good approximation. Finally, four properties that are used in a design procedure established in the next section are postulated.

3.1 Comparison of results with independence assumption

In the engineering of automatic call distributor systems, a traffic engineer generally dimensions each trunk group using the Erlang loss formula (assuming that it is independent of the other groups and the number of attendants) and often determines the number of attendants required from the Erlang delay formulas using the total offered load to the queues (assuming no blocking in the queues). This procedure is clearly invalid, but up to now an exact procedure has not been available. As a means of illustrating the significance of the results presented in this paper, we now compare, for a particular system configuration, the system characteristics that a traffic engineer would expect to obtain using the independence assumption and what he really will find. Of course, the interqueue service discipline will affect these results in a way that will be described in later work. The actual operation of such systems is quite complicated, and is not readily characterized by any of the disciplines usually used such as first-in, first-out, random, and last-in, first-out. However, the results of simulations indicate that the discipline presented here is a good approximation of the actual method of operation.

For purposes of this comparison, we assume a simple system with only two queues: the first with an offered busy-hour load of 10 erlangs; the second, 5 erlangs. It is further assumed that the queues have coincident busy hours and that the average holding time per call is 30 seconds. Assuming that a P.01 grade-of-service constraint has been placed on each group, the number of trunks required, if independence is assumed, would be $N_1 = 18$ and $N_2 = 11$. (These numbers can be found from tables of the Erlang B formula.) If it is then required that, on the average, no call must wait longer than 3 seconds for an answer, we find, from the Erlang delay formula, that $M = 19$ (assuming no

blocking in the groups). The traffic engineer would expect this system to have the following characteristics:

Blocking probability for group 1 = 0.0071.
Blocking probability for group 2 = 0.0083.
Overall average delay = 1.83 seconds.

However, analyzing this system configuration with the results of this paper, we find that the service levels (which stated above are approximations to the actual levels) would be

Blocking probability for group 1 = 0.014.
Blocking probability for group 2 = 0.013.
Overall average delay = 0.949 second.

The directions of the changes are intuitively obvious, since longer holding times in the queues result in more blocked calls and the higher blocking levels decrease the load to the servers, which in turn results in a lower average delay. It should be noted that the trunk groups are performing at unsatisfactory levels, but the overall average delay has been decreased and is considerably under the required level. Often, customers who have such systems measure only the delay or speed of answer and periodically remove attendants if they feel that the speed of answer is not above the required level. Unfortunately, such a customer generally does not realize the effect of removing attendants on the blocking probabilities on his incoming trunks and consequently on other network customers.

As an example of customer behavior, consider the system discussed above. The customer, having measured the average delay and finding it to be considerably under his required level, would most likely remove two attendants. The average delay would then become 3.01 seconds, but the blocking probabilities increase to 0.028 for group 1 and 0.023 for group 2. Hence, even though the delay constraint is essentially satisfied, the probabilities of loss are more than double their desired levels.

If, instead of the P.01 service level, P.05 or P.10 had been chosen for the above delay constraint, the independence assumption would generally give a configuration that would satisfy all service constraints. The reason for this is that the higher blocking levels decrease the offered load to the attendants, and consequently a very small delay results. This delay is small enough that it has little effect on the holding time of the calls and, hence, the offered load to queue i is approximately a_i. Therefore, the resulting blocking probability, although larger than the Erlang loss probability, is generally in the acceptable

range. However, the configuration would not be optimal because too many attendants are provided. In summary, as the blocking probabilities increase, the discrepancies between the Erlang loss formula and the formula given by (18) decrease, but the discrepancies between average delays increase.

Several variations in the procedures to engineer these systems exist in practice. Generally, in determining the number of attendants, the *measured* offered load into the attendants is used. This load accounts for the blocking in each trunk group. For the above example, the measured offered load would be 14.79 erlangs (if we assume that the assumptions made in this analysis are valid) and, from the Erlang delay formula, an average delay of 1.58 seconds results, which is still higher than the actual delay. The reason for the discrepancies is that the offered load to the attendants is no longer Poisson as a result of the blocking in the groups. In fact, the variance of this offered load will be lower than that of the Poisson load, since the peakedness of the traffic has been decreased by clipping. Hence, since the actual average offered load and variance of the load are lower, this leads us to postulate the following property:

Property 1:

$$\bar{W}(M, a) \geqq \bar{D}(\mathbf{N}, M, \mathbf{a}) \quad \text{where} \quad a = \sum_{i=1}^{l} a_i.$$

This property is illustrated in Fig. 1. The equality holds in the limit as $N_i \to \infty$ for all $i = 1, \cdots, l$, as shown previously. The significance of this property is that we now have a method of obtaining an upper bound on the average delay for the combined system.

Another variation that is sometimes used is to add the speed of answer into the offered load to each group. If we add the average delay of 0.949 second to each call and use this new offered load in the Erlang loss formula, the blocking probabilities that result are 0.0091 and 0.01, for groups 1 and 2, which are still lower than the actual blocking.

The discrepancies result because the Erlang loss formula assumes exponential holding times on the trunks but, in fact, the holding time for the combined system is the sum of an exponential distribution and the delay distribution. Also, the holding times of calls in the system are no longer independent (unless $\sum_{i=1}^{l} n_i < M$). The variance of this new service time distribution is higher than that of the exponential service time distribution and, of course, the mean is larger. Therefore, one would expect the average queue length to be larger which, in turn, implies an increase in blocking.

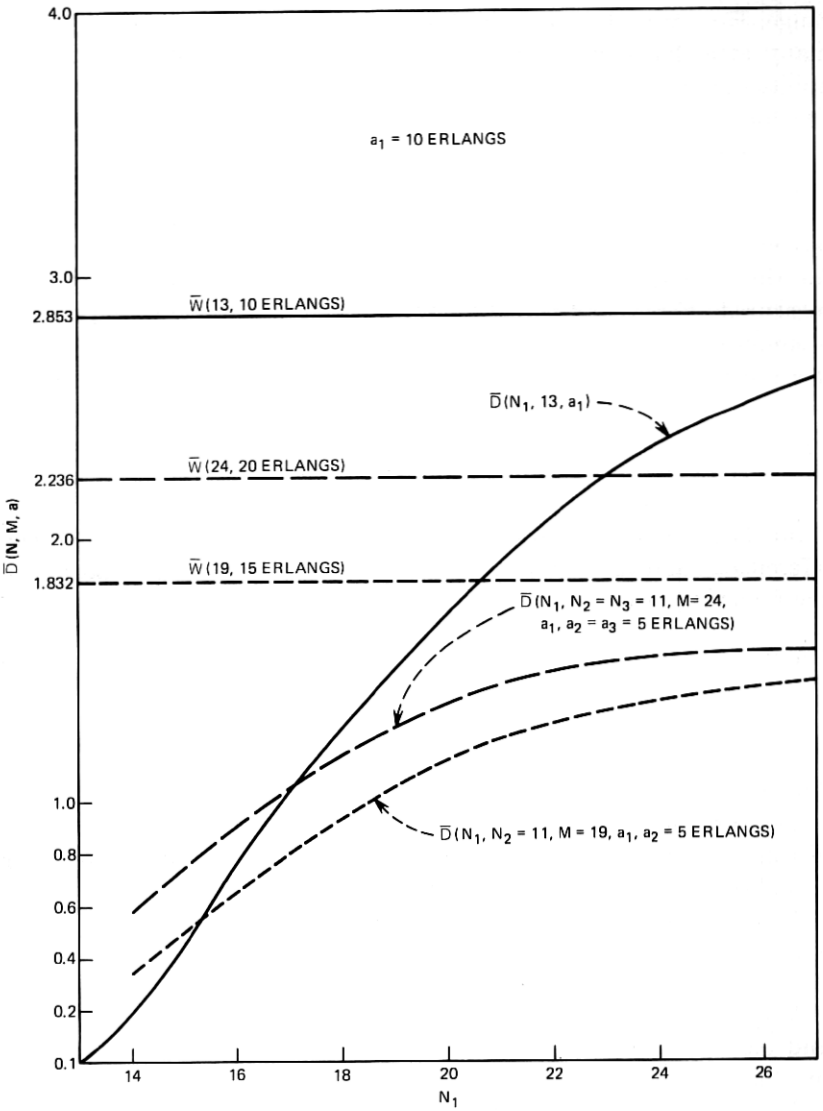

Fig. 1—Average delay as a function of number of positions in one queue.

With these facts in mind and as a result of empirical evidence, we postulate a second useful property:

Property 2:

$$B(N_i, a_i) \leqq B_i(\mathbf{N}, M, \mathbf{a}) \qquad (i = 1, \cdots, l),$$

where $B(N_i, a_i)$ is the Erlang loss formula for N_i servers and an offered

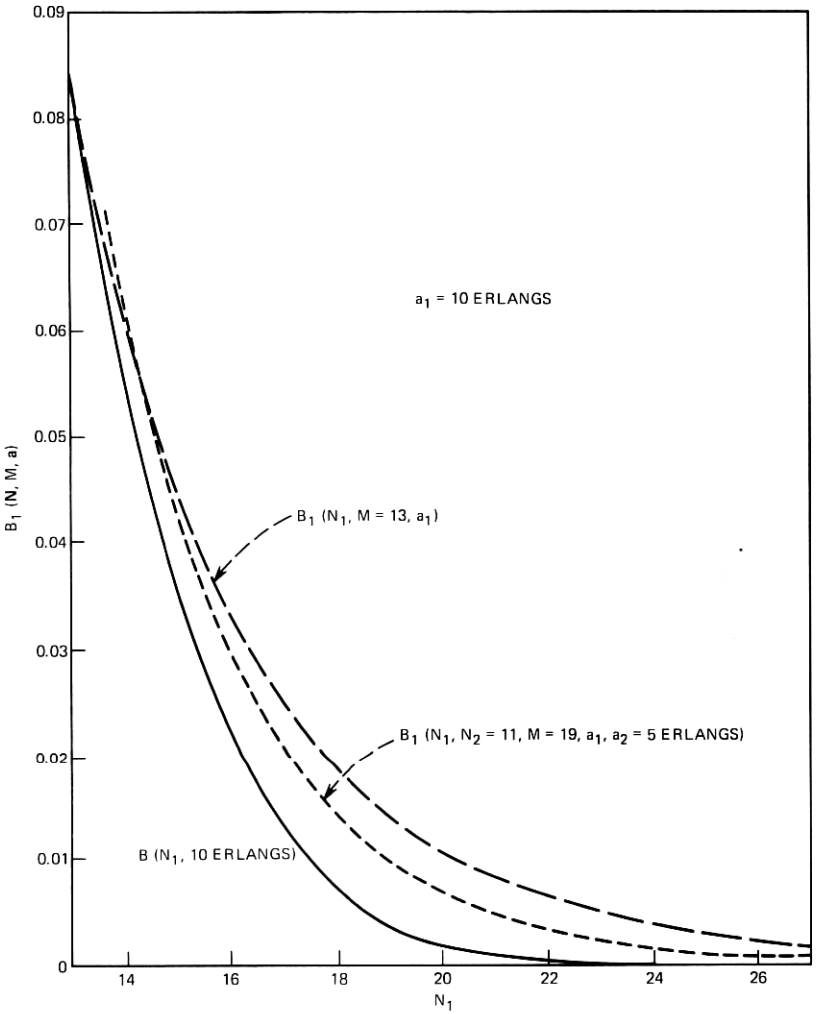

Fig. 2—Blocking probability as a function of number of positions in the queue.

load, a_i. The equality holds when $M \geqq \Sigma N_i$, since the queues then behave independently (there is no delay, so in fact the holding time per call is exponential). This property is illustrated in Fig. 2. Intuitively, one would expect $B_i(\mathbf{N}, M, \mathbf{a})$ to be larger since, as a result of a positive delay added to each call, calls hold the trunks longer, therefore increasing the probability of an arriving call finding all trunks busy. The significance of Property 2 is that a lower bound on the number of trunks required for a given service level can be found using the Erlang loss formula.

3.2 Effects of varying N_i and M

For a given input process and service time distribution, the designer can affect the blocking probabilities for a queue or the average delay by changing the number of positions, N_i, in a queue and/or changing the number of servers, M. We first investigate the effects of varying the number of servers, M. As noted earlier, when M has been increased to the point where $\sum_{i=1}^{l} N_i = M$, the average delay becomes zero and the queues behave independently. The blocking probability for each queue is then given by $B(N_i, a_i)$ which, by Property 2, is a lower bound on the blocking for any value of M. What is of importance, however, is: Do the blocking probabilities and the average delay monotonically decrease to their lower bounds as we increase M to the value $\sum_{i=1}^{l} N_i$? Empirical evidence, such as shown in Figs. 3 and 4, indicates that the answer to this question is *yes*. We postulate this property as:

Property 3:

$$B_i(\mathbf{N}, M+1, \mathbf{a}) \leqq B_i(\mathbf{N}, M, \mathbf{a}) \qquad (i = 1, \cdots, l)$$
$$\bar{D}(\mathbf{N}, M+1, \mathbf{a}) \leqq \bar{D}(\mathbf{N}, M, \mathbf{a}).$$

Intuitively, one would not expect that increasing the number of servers in a system would increase the average delay. Moreover, a decrease in the holding time of calls would imply that the average queue lengths would decrease, which would result in a decrease in the blocking probability for that queue. However, this decrease in blocking results in an increase in offered load to the servers but, as we postulate, this increase is less than the marginal carrying capacity of the added server. The significance of this property is that, with added servers, not only is the average delay decreased but also the blocking probabilities are decreased; that is, adding servers improves the service performance of the servers *and* of the queues (trunk groups).

The other system parameters that may be varied to improve system performance are the numbers of positions in each queue. Supported by quantitative evidence, such as given in Figs. 1 and 5, and by intuition we postulate the following:

Property 4:

$$B_i(\mathbf{N}_{i+}, M, \mathbf{a}) \leqq B_i(\mathbf{N}, M, \mathbf{a})$$
$$B_j(\mathbf{N}_{i+}, M, \mathbf{a}) \geqq B_j(\mathbf{N}, M, \mathbf{a}) \qquad j \neq i.$$
$$\bar{D}(\mathbf{N}_{i+}, M, \mathbf{a}) \geqq \bar{D}(\mathbf{N}, M, \mathbf{a})$$

The first part of this property states that, if we increase the number of positions for calls to occupy in a given queue, then the probability of

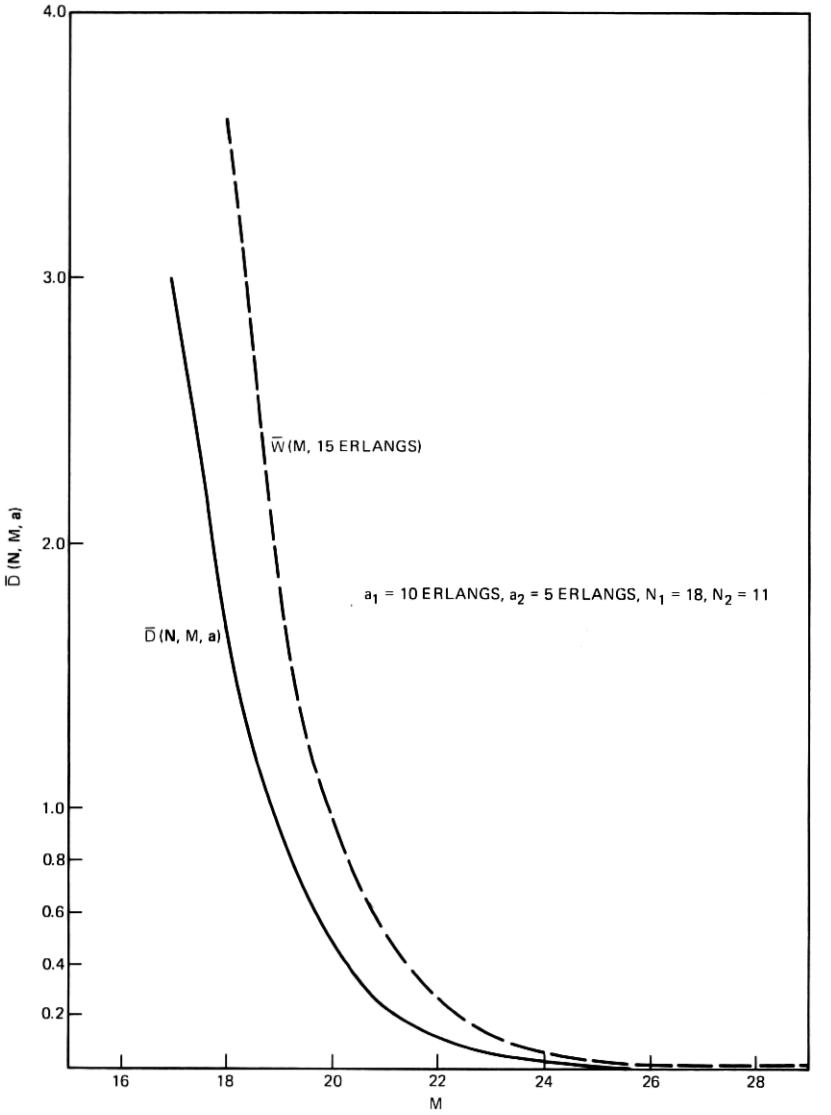

Fig. 3—Average delay as a function of number of servers.

loss for that queue is decreased. (This, of course, is true in pure loss systems.) The intuitive argument is that calls that previously found N_i calls in queue i were blocked, but now are not. Therefore, the number of calls blocked is decreased. However, as the third part of this property implies, the average delay of calls is increased as a result of this increase in calls from queue i. The intuitive counter-

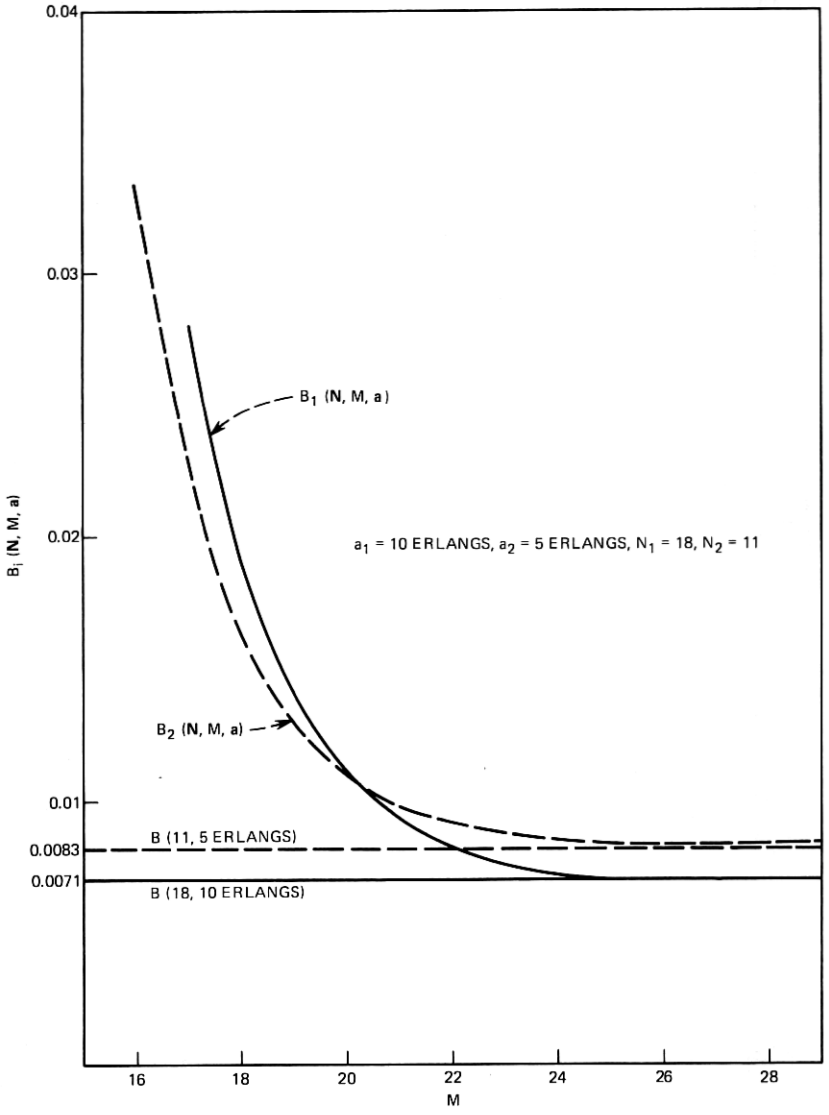

Fig. 4—Blocking probability as a function of number of servers.

argument is that this increase in holding time per call might result in an increase in blocking for queue i. But we postulate that the increase in delay is not substantial enough to eliminate the increased efficiency obtained in queue i by the addition of a position.

However, for the other queues, the number of positions remains fixed, and this increase in average delay results in a larger traffic level

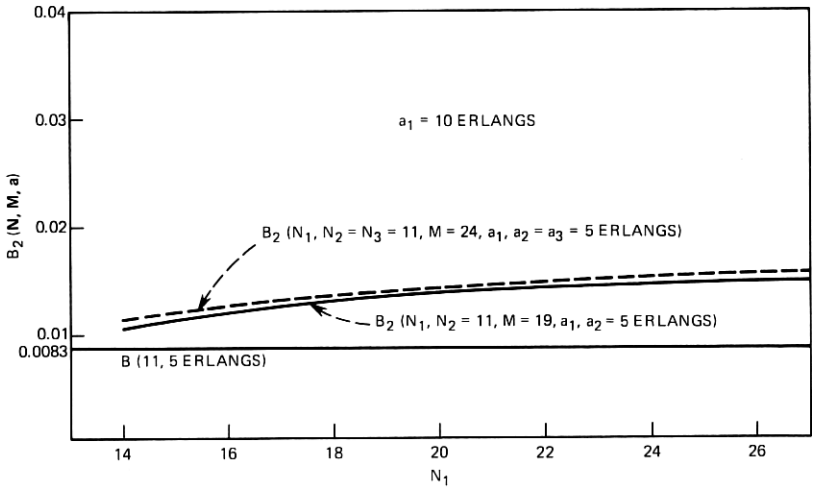

Fig. 5—Blocking probability in one queue as a function of number of positions in another.

per position being placed on the queues. Hence, this increase in load causes the average queue length to increase and, consequently, causes the blocking probabilities to increase. This fact is stated in the second part of Property 4 and is illustrated by the example given in Fig. 5. Equality in the three statements holds only in the limit as N_i goes to ∞ for all $i = 1, 2, \cdots, l$.

The significance of Property 4 is that the loss probability for a given queue cannot be reduced by adding positions to any other queue. Therefore, by Properties 3 and 4, we see that the loss probability for a given queue can be reduced only by increasing the number of servers or by increasing the number of positions in that queue.

The four properties postulated indicate relationships between the system parameters and the system characteristics. Proofs for the simplest cases (i.e., $l = 1$ for all the properties except for the second part of Property 4 for which $l = 2$) are given in Ref. 18. The proofs for the general cases have not been constructed because of the difficulties involved (e.g., the proof of the second part of Property 4 required 17 pages for $l = 2$). However, based on the intuitive explanations given, empirical evidence, and these proofs, I feel that the properties are valid in the general case of l queues. To obtain an optimal configuration (minimum cost), one must balance the cost of servers against the cost of positions for the queues in such a way that all required service levels are met. These properties are used in the next section in the development of a procedure to determine this optimal

configuration. The algorithm and proof of convergence given below assume the validity of these properties.

IV. SYSTEM DESIGN WITH GRADE-OF-SERVICE CONSTRAINTS

4.1 The design problem

In determining the optimal design of a queuing system, one is generally interested in minimizing the operating costs in such a way that specified grade-of-service constraints are met. These constraints are a function of the particular queuing system under study. In the pure loss system, the grade of service is measured by the probability that a call is lost or blocked. Hence, the optimal system configuration is the minimum number of servers that satisfies the constraint, $B(s, a) \leqq b$.

In delay systems, at least three measures of service are useful. The first is the average delay experienced by a call in obtaining service. The second measure is the "extremal" delay—the probability that the delay for any call exceeds a specified limit. Finally, the average delay of only those customers who experience some delay is a useful measure.

However, in the combined loss and delay systems described in Section II, the determination of an optimal configuration is not as straightforward. We will measure the grade of service of the system by

(*i*) The blocking probability for each queue.
(*ii*) The average delay of all calls that enter the system.

The blocking probability for a particular queue is dependent on the number of positions in each queue and the number of servers, and the average delay depends on these same variables. A procedure must be developed to balance these measures of congestion in such a manner that the costs of the system are minimized.

More formally, the problem can be expressed as the following nonlinear problem in integer programming. We denote the monthly cost of a waiting position in queue i by C_i and the monthly cost of each server by C. It is assumed that C_i and C are positive, finite numbers. The blocking objective for queue i will be denoted by b_i and the average delay objective by d. The following assumptions have been made: The system is engineered for the system busy-hour traffic load and the busy hours for the queues are coincident. The optimization problem is then expressed as:

Minimize the cost

$$Z = \sum_{i=1}^{l} C_i N_i + CM$$

subject to

$$\text{(I)} \qquad B_i(\mathbf{N}, M, \mathbf{a}) \leqq b_i \qquad (i = 1, 2, \cdots, l) \tag{21}$$

$$\bar{D}(\mathbf{N}, M, \mathbf{a}) \leqq d \tag{22}$$

$$N_i (i = 1, 2, \cdots, l), \qquad M \geqq 0 \text{ integers.}$$

Since no efficient algorithm to solve a general nonlinear integer-programming problem exists, a procedure was developed that utilizes the four properties stated in Section III.

4.2 Optimization procedure

In this section, a procedure is developed that determines an optimal solution to the nonlinear integer-programming problem expressed by (I). The procedure is a direct-search routine based primarily on the properties presented in the previous section. A description of the procedure is now given and is followed by a concise summary of the algorithm. Figure 6 is a flowchart of the algorithm.

The first step in the algorithm, as in most mathematical programming algorithms, is the determination of a feasible solution to the problem. To obtain an initial feasible solution to (I), we utilize Properties 2 and 3 of the previous section. In particular, by Property 2, we know that the minimum number of waiting positions for queue i can be determined from the Erlang loss formula, which is easily computed from a recurrence relationship.[19] We begin the search for an initial feasible solution with

$$N_i^{(0)} = \min \{n \mid B(n, a_i) \leqq b_i\},$$

since it has been shown that, in fact, a feasible solution to (I) exists. That is, $(\mathbf{N}^{(0)}, \tilde{M})$, where $\tilde{M} = \sum_{i=1}^{l} N_i^{(0)}$, is a feasible solution since $\bar{D}(\mathbf{N}^{(0)}, \tilde{M}, \mathbf{a}) = 0$ and $B_i(\mathbf{N}^{(0)}, \tilde{M}, \mathbf{a}) = B(N_i^{(0)}, a_i)$ for $i = 1, 2, \cdots, l$. However, since this solution will generally not be near the optimal solution, the search will not begin at $\tilde{M}$ but instead with $M_{(0)}$, which is the minimum value of M that satisfies the constraint:

$$\bar{W}(M, a) \leqq d. \tag{23}$$

$M_{(0)}$ is the number of servers that would be selected if the Erlang delay formula with an offered load $a = \sum_{i=1}^{l} a_i$ were used. By Property 1, it is seen that if (23) is satisfied, then (22) will also be satisfied.

Using the set of parameters $(\mathbf{N}^{(0)}, M_{(0)})$, the system characteristics, $\{B_i(\mathbf{N}^{(0)}, M_{(0)}, \mathbf{a})\}$ and $\bar{D}(\mathbf{N}^{(0)}, M_{(0)}, \mathbf{a})$ are determined. One of two results occurs:

(*i*) The parameters ($\mathbf{N}^{(0)}$, $M_{(0)}$) satisfy the constraints of (I), in which case an initial feasible solution has been determined. We then proceed to find the feasible solution of minimum cost.

(*ii*) At least one of the blocking constraints (21) is violated. (As noted above, the delay constraint will be satisfied.) By Property 3, we know that the addition of a server will reduce the blocking probability for each queue and that, by the addition of enough servers, a feasible solution can be obtained. We denote this initial feasible solution by ($\mathbf{N}^{(0)}$, $M^{(0)}$) and note two interesting properties of this solution.

(*a*) By Property 2, $N_i^{(0)}$ is the *minimum* number of positions that must be considered for queue i.

(*b*) $M^{(0)}$ is the *maximum* number of servers that must be considered. This is true since a further increase in the number of servers can be justified only if some waiting positions can be eliminated, and the positions are already at their minimum levels, $\mathbf{N}^{(0)}$.

The next step in the algorithm is to attempt to improve the initial feasible solution. Since by construction we are initially at the maximum number of servers, we attempt to decrease the costs by decreasing the number of servers while maintaining feasiblity. To maintain feasibility, it may be necessary to add waiting positions to certain queues. If a feasible solution is found, then it will be an improvement only if the accumulated cost of those servers removed (since the last feasible solution) is greater than the accumulated cost of all waiting positions that have been added to maintain feasibility. Hence, as we remove servers, one of three things results.

(*i*) All the constraints of (I) are satisfied. If the accumulated cost of removed servers is greater than the cost of all waiting positions that have been added, this new feasible solution represents a cost improvement and should be stored as the tentative "optimal" solution. The accumulated costs are set to zero, a server is removed, and the search continues. If the cost of servers is less than the cost of positions, then we reduce the number of servers by one and continue the search for a solution with lower cost.

(*ii*) The delay constraint (22) is violated. In this case, we stop the search and the tentative optimal solution is the global* optimum. (A justification for stopping the procedure is given later.)

(*iii*) At least one of the blocking constraints is violated. In this case, we add one position to each queue in which the corresponding block-

---

*I have taken the liberty of using "global" since, in fact, the procedure does produce the global optimum *if* the four properties are true in the general case.

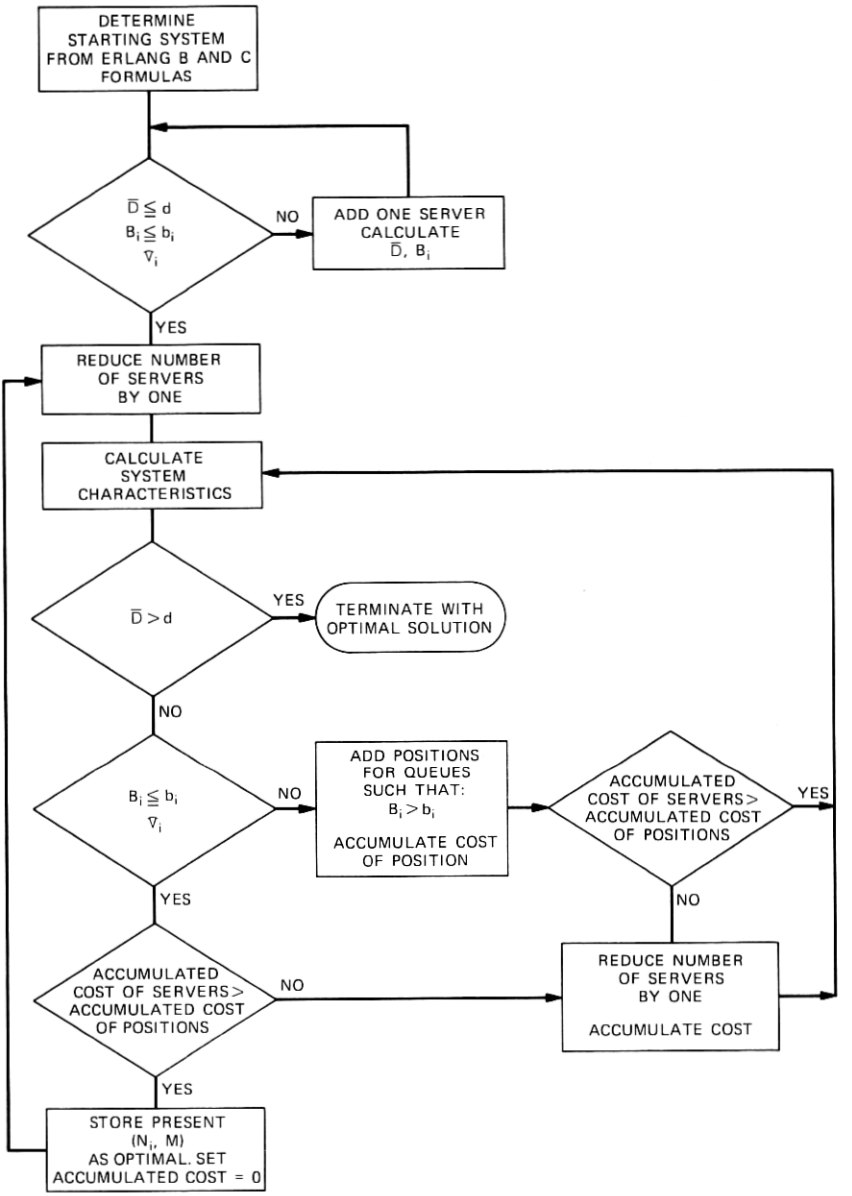

Fig. 6—Design procedure.

ing constraint is violated and increase the accumulated cost of added positions appropriately. If the cost of the additional positions is less than the accumulated cost of the servers that have been removed, we determine if this solution is feasible. If it is feasible, we proceed as in

(*i*). If it is not feasible, either (*ii*) or (*iii*) must be true. If the cost of the positions is more than the cost of servers, then no feasible solution with lower cost can be obtained with this number of servers and, hence, we reduce the number of servers by one and calculate the system characteristics with the new set of system parameters. Then either (*i*), (*ii*), or (*iii*) must be true, and the appropriate action is then taken.

In summary, the procedure removes servers until either the delay constraint is violated, in which case it terminates, or a blocking constraint is violated. If a blocking constraint is violated, waiting positions are added, and an additional server *may* be removed, depending on the incremental costs and on the feasibility of the tentative solution.

A justification of the procedure is in order. We first discuss the case in which at least one blocking constraint has been violated. To reduce the blocking probability for each queue whose constraint has been violated, a position must be added to this queue (by Property 4). The only other way to reduce the blocking probability is to add a server, but this branch has already been terminated. Assume that NS servers have been removed since the last "optimal" feasible solution and that the cost of all the positions added since the last "optimal" feasible solution is G. If $G > NS \times C$, then the new parameters cannot give a lower cost solution and, by Property 4, no lower cost solution for this M exists. Therefore, we terminate the branch with M servers and begin the search of the branch with $M - 1$ servers (increment NS by 1) with the present number of positions. At least this number of positions must be considered, since, by Property 3, the reduction of M results in an increase in the blocking probabilities. If $G \leqq NS \times C$, this set of parameters may be a lower-cost solution. Therefore, we determine the system characteristics and see if the solution is feasible.

We now prove that the procedure converges in a finite number of steps to a global optimum.

Lemma 1: *If a feasible solution for a given M has been found, the branch corresponding to that M can be terminated.*

Proof: Trivially, any further feasible solutions with that M must be more expensive, since these solutions must have more waiting positions. Q.E.D.

Theorem 1: *The algorithm terminates in a finite number of steps.*

Proof: First, we know that an initial feasible solution can be obtained in at most

$$\left[\sum_{i=1}^{l} N_i^{(0)} - M_{(0)} \right]^+$$

steps, as discussed earlier, where $[x]^+ = \max(0, x)$. Second, for any given M, the corresponding branch of the solution tree will be terminated in a finite number of steps. That is, since C is finite and C_i is nonzero for all i, then in a finite number of steps we will either find a feasible solution for this M, reach the point where the cost of the positions added for this M exceeds C, or violate the delay constraint. In the first case, by Lemma 1, we know that we can terminate this branch. In the second case, since none of the positions can be removed and feasibility be maintained, no feasible solution of lower cost exists for this M. In the latter case, the procedure terminates. The number of iterations performed for a given M is bounded by the number of times positions are added before the cost of these additions exceeds C.

Finally, since the maximum number of servers that need be considered is finite, we reach the case in which the delay constraint is violated in a finite number of steps (at most, $M^{(0)}$ values of M).

Since there are only a finite number of values of M to be considered and since, for each M, only a finite number of steps are performed, the algorithm terminates in a finite number of iterations. Q.E.D.

Theorem 2: *The solution* $(\mathbf{N}^*, M^*, Z^*)$ *obtained upon termination of the algorithm is a global optimum.*

Proof: Assume that another configuration $(\hat{\mathbf{N}}, \hat{M}, \hat{Z})$ exists, such that all constraints of (I) are satisfied and $\hat{Z} < Z^*$. First, consider the case in which $\hat{M} > M^*$. By construction, the branch corresponding to $\hat{M}$ must have been searched and, as indicated in Theorem 1, the branch would have been terminated in a finite number of steps. If this branch had produced a feasible solution with a lower cost, it would have been retained. Hence, this case is not possible.

Second, consider the case in which $\hat{M} < M^*$. From the algorithm, we know that the termination of the procedure implies that the delay constraint has been violated. We therefore know that either the branch with $\hat{M}$ produced a feasible solution with a cost larger than Z^* (or else it would have been retained), or $\hat{M}$ is smaller than the value of M when the procedure terminates. If the latter is true, then $\hat{M}$ cannot produce a feasible solution since the delay and blocking probability for $(\mathbf{N}^*, \hat{M})$ must be greater than those for $(\mathbf{N}^*, M^*)$ by Property 3; and, to reduce this delay, positions would have to be removed that would result in at least one blocking constraint being violated. Consequently, this case is a contradiction and, hence, $(\mathbf{N}^*, M^*, Z^*)$ is the optimum. Q.E.D.

One point should be noted: For the higher levels of blocking ($\gtrsim 0.05$), the solution $(\mathbf{N}^{(0)}, M_{(0)})$ is generally a feasible solution. However, as a result of the reduction in offered load to the servers because of the

blocking, $M_{(0)}$ can be reduced before any constraints are violated. This is of importance since, in practice, such systems have been engineered in such a manner that the configuration is $(\mathbf{N}^{(0)}, M_{(0)})$, which is obtained from use of the Erlang B and Erlang C formulas, as described earlier.

A more concise mathematical summary of the algorithm follows.

Optimization Algorithm:

Step 1: Initial Feasible Solution

(*i*) Determine $N_i^{(0)}$ $(i = 1, 2, \cdots, l)$ from the Erlang loss formula where $N_i^{(0)} = \min \{n | B(n, a_i) \leqq b_i\}$.

(*ii*) Determine $M_{(0)}$ from the Erlang delay formula where $M_{(0)} = \min \{m | \bar{W}(m, a) \leqq d\}$. Let $k = 1$.

kth iteration:

(*iii*) Calculate $B_i(\mathbf{N}^{(0)}, M_{(k-1)}, \mathbf{a})$ and $\bar{D}(\mathbf{N}^{(0)}, M_{(k-1)}, \mathbf{a})$. If the set of constraints (21) are satisfied, let $M^{(0)} = M_{(k-1)}$. The initial "optimal" feasible solution is $\mathbf{N}^* = \mathbf{N}^{(0)}$, $M^* = M^{(0)}$, and $Z^* = \sum_{k=1}^{l} C_i N_I^* + CM^*$. Set $j = 1$, ${}_1N^{(0)} = N^*$, and go to Step 2.

If at least one constraint of (21) is not satisfied, set $M_{(k)} = M_{(k-1)} + 1$, increment k, and return to (*iii*).

Step 2: Solution Improvement

In this step, the superscript j refers to the jth "optimal" value of M; for a given value of j, the subscript r refers to the rth value of N_i.

jth iteration:

(*i*) $NS = 1$, $G^{(j)} = 0$, $r = 1$.

(*ii*) Reduce number of servers by one, $M^{(j)} = M^{(j-1)} - 1$. If $M^{(j)} = 0$, go to Step 3.

(*iii*) Calculate $B_i({}_r\mathbf{N}^{(j-1)}, M^{(j)}, \mathbf{a})$ and $\bar{D}({}_r\mathbf{N}^{(j-1)}, M^{(j)}, \mathbf{a})$.

(*a*) If $\bar{D}({}_r\mathbf{N}^{(j-1)}, M^{(j)}, \mathbf{a}) > d$, go to Step 3.

(*b*) If (21) and (22) are satisfied and if $G^{(j)} \leqq NS \times C$, then store $({}_r\mathbf{N}^{(j-1)}, M^{(j)})$ as the new "optimal" solution. That is, set $\mathbf{N}^* = {}_r\mathbf{N}^{(j-1)}$, $M^* = M^{(j)}$ and

$$Z^* = \sum_{i=1}^{l} C_i \, {}_rN^{(j-1)} + CM^{(j)}.$$

Set ${}_1\mathbf{N}^{(j)} = {}_r\mathbf{N}^{(j-1)}$, increment j, and go to (*i*).

(*c*) If (21) and (22) are satisfied but $G^{(j)} > NS \times C$, then reduce the number of servers by subtracting 1 from $M^{(j)}$, increment NS, and return to (*iii*).

(*d*) If at least one blocking constraint of (21) is violated, i.e., $S^{(j)} = \{i | B_i({}_r\mathbf{N}^{(j-1)}, M^{(j)}, \mathbf{a}) > b_i\}$ is not empty, then add $\sum_{i \in S^{(j)}} C_i$ to $G^{(j)}$. Let

$$ {}_{r+1}N_i^{(j-1)} = {}_rN_i^{(j-1)} \qquad i \notin S^{(j)} $$

and

$$ {}_{r+1}N_i^{(j-1)} = {}_rN_i^{(j-1)} + 1 \qquad i \in S^{(j)}. $$

If $G^{(j)} > NS \times C$, then decrement $M^{(j)}$ by 1, increment r, and go to (*iii*). If $G^{(j)} \leqq NS \times C$, increment r, and go to (*iii*).

Step 3: Termination

If $\bar{D}({}_r\mathbf{N}^{(j-1)}, M^{(j)}, \mathbf{a}) > d$, then stop the procedure. The global optimum is $(\mathbf{N}^*, M^*, Z^*)$.

4.3 Numerical example

To illustrate the algorithm, we examine a simple two-queue system that represents an automatic call distributor system used for credit checking. The company has subscribed to two Inward WATS bands with a cost per trunk of \$800 and \$500. The two trunk groups each receive 15 erlangs of traffic in the busy hour. Calls, on the average, are 45 seconds in duration. The subscriber has requested a 5-second speed of answer and blocking objectives of P.10 and P.05, respectively. The monthly cost of an attendant is \$750. With these parameters, we begin the algorithm by using the Erlang loss formula with 15 erlangs and the delay formula with 30 erlangs to obtain $(\mathbf{N}^{(0)}, M_{(0)})$, which are (18, 20; 34). As shown in Table I, this initial solution is feasible and hence will be stored as our tentative optimal solution, $(\mathbf{N}^{(0)}, M^{(0)})$.

We proceed to Step 2 of the algorithm in an attempt to improve the initial feasible solution. By decreasing $M^{(0)}$ by one, $M^{(1)} = M^{(0)} - 1$, we obtain the system parameters (18, 20; 33) and the system characteristics (0.0912, 0.0504; 0.404), which indicate that this is not a feasible solution. Since the blocking constraint for trunk group 2 is violated, a trunk must be added to this group at a cost of \$500. The accumulated cost of the additional trunks, \$500, is less than the accumulated cost of the removed attendants, \$750. Therefore, we proceed by obtaining the system characteristics for this set of param-

Table I — An example of the optimization procedure

		N_1	N_2	M	B_1	B_2	$\bar{D}$	$Z(\$)$
Initial Feasible Solution	1	18	20	34	0.0887	0.0481	0.201	49900.
Solution Improvement	2	18	20	33	0.0912	0.0504	0.404	49150.
	3	18	21	33	0.0926	0.0373	0.546	49650.
	4	18	21	32	0.0970	0.0409	0.945	48900.
	5	18	21	31	0.1034	0.0460	1.550	48150.
	6	19	21	30	0.0939	0.0573	2.990	48200.
	* 7	19	22	30	0.0976	0.0466	3.422	48700.
	8	19	22	29	0.1120	0.0572	4.976	47950.
Termination	9	20	23	28	0.1220	0.0676	8.864	48500.

* Optimal solution.
System Parameters:

$a_1 = 15$ erlangs, $b_1 = 0.10$, $c_1 = \$800$
$a_2 = 15$ erlangs, $b_2 = 0.05$, $c_2 = \$500$
$HT = 45$ s, $d = 5$ s, $C = \$750$.

eters, i.e., (18, 21; 33). As Table I indicates, this is a feasible solution, is a cost improvement, and hence is stored as the tentative optimum.

As shown in Table I, the procedure continues from this point until (20, 23; 28), at which point the delay constraint is violated. The optimal solution is (19, 22; 30) at a cost of \$48,700. We should note that the parameters (20, 23; 29) were not examined since the accumulated cost of added trunks, \$1300, was greater than the accumulated cost of removed attendants, \$750.

The above solution is the global optimum for this constrained problem. However, practitioners might suggest that (18, 21; 31) is a more realistic design since, in fact, the blocking constraint is "essentially" satisfied (0.1034 vs 0.1000). This can be incorporated in the design procedure by allowing any solution that is within "ϵ" of a blocking objective to be retained. The algorithm can then be applied as before.

V. SUMMARY

In this paper, an analysis of a particular multiserver, multiqueue service system has been presented. Examples of this type of system are the directory assistance systems used in the telephone companies and credit verification bureaus used by the credit-card industry, which use automatic call distributors. Expressions were derived for the equilibrium-state probabilities, and four properties of the system characteristics, overall average delay, and the blocking probabilities for each queue were given.

These results were used in developing a procedure to obtain a least-cost system configuration to satisfy a given set of single-hour grade-of-

service constraints. That is, the procedure determines the number of waiting positions for each queue and the number of servers required to satisfy constraints placed on blocking probabilities and average delay at minimum cost. The work reported here should form the basis for the development of a practical method of traffic engineering and administration for small automatic call distributor systems.

REFERENCES

1. E. Brockmeyer, H. A. Halstrom, and A. Jensen, *The Life and Works of A. K. Erlang*, Trans. 2. Copenhagen: Danish Academy of Technical Sciences, 1948.
2. K. Lundkvist, "General Theory for Telephone Traffic," Ericsson Technics, *9*, No. 2, 1953, pp. 111–140.
3. H. Störmer, "Wartezeitlenkung in Handbedieten Vermittlungsanlagen," Archiv für Electronik und Übertragungstechnik, *10*, No. 2 (February 1956), pp. 58–64.
4. J. R. W. Smith and J. L. Smith, "Loss and Delay in Telephone Call Queueing Systems," A.T.E. Journal, *18*, No. 1 (January 1962), pp. 18–30.
5. J. Riordan, *Stochastic Service Systems*, New York: John Wiley, 1962, pp. 96–101.
6. L. Takács, "On a Combined Waiting Time and Loss Problem Concerning Telephone Traffic," Ann. Univ. Sci. Budapest Eötvös, Sect. Math., *1*, 1958, pp. 73–82.
7. A. Descloux, "On Markovian Servers with Recurrent Input," Sixth International Teletraffic Congress, Munich, 1970.
8. W. C. Chan, "Combined Delay- and Loss-Queueing System," Proc. IEE, *117*, No. 11 (November 1970).
9. G. Basharin, "Servicing Two Flows in a Single Queue System with a Limited Number of Places for Waiting and Absolute Priority," Eng. Cybernetics, *15*, Oct. 1967, pp. 95–105.
10. W. Wagner, "On Combined Delay and Loss Systems with Nonpreemptive Priority Service," Fifth International Teletraffic Congress, New York, 1967.
11. J. Brandt, "A Multiserver Queueing System with Preemptive Priority," Sixth International Teletraffic Congress, Munich, 1970.
12. P. Kühn, "Parallel Waiting Queues in Real-Time Computer Systems," Nachrichtentech. Z., *23*, No. 11 (November 1970), pp. 576–582.
13. P. Kühn, "Combined Delay and Loss Systems with Several Input Queues, Full and Limited Accessibility," Sixth International Teletraffic Congress, Munich, 1970.
14. P. Kühn, "Combined Delay and Loss Systems with Several Input Queues, Full and Limited Accessibility," Archiv für Electronik und Übertragungstechnik, *25*, No. 10–11 (September-October 1971), pp. 449–454.
15. R. Syski, *Introduction to Congestion Theory in Telephone Systems*, London: Oliver & Boyd, 1960, pp. 254.
16. J. D. C. Little, "A Proof for the Queueing Formula: $L = \lambda W$," Operations Research, *9*, No. 3 (May-June 1961), pp. 383–387.
17. R. Morris and E. Wolman, "A Note on 'Statistical Equilibrium,'" Operations Research, *9*, No. 5 (September-October 1961), pp. 751–753.
18. B. A. Whitaker, "An Analysis and Optimization of Multiserver, Multiqueue Systems with Finite Waiting Space in Each Queue," Ph.D. Dissertation, New York University, 1974.
19. R. B. Cooper, *Introduction to Queueing Theory*, New York: Macmillan, 1972, p. 100.