Journal of the Operations Research
Society of Japan
Vol. 29, No. 4, December 1986

APPROXIMATE ANALYSIS OF AN ON-LINE NETWORK SYSTEM

Huanxu Pan Hidenori Morimura
Tokyo Institute of Technology

(Received September 18, 1985; Final September 25, 1986)

Abstract This paper provides an approximate analysis for an on-line network system consisting of a CPU and a large number of terminals located in many service stations. Customers arrive at each station for CPU service via a Poisson process. The CPU processes jobs from terminals according to the processor sharing discipline. The CPU processing rate is dependent on the number of customers receiving the CPU service simultaneously and decreases when this number increases beyond the CPU capacity. The approximate analysis in this paper is performed by dividing the entire system into a computer subsystem and queueing subsystems. The computer subsystem is formulated as a closed queueing network and each queueing subsystem as an $M/G/m$ queue. Our results are simple and can be applied easily to decide on the optimal number of terminals that minimizes mean total system time.

Key Words: on-line network system; closed queueing network; $M/G/m$ queue; terminal control policy.

1. Introduction

Nowadays we can see a vast use of so called information network systems such as cash dispenser service systems in banks and on-line information retrieval service systems in libraries. In such a service system, a huge computer deals with service requirements from hundreds or thousands of terminals located in various places. We call such a service system as an *on-line network system.*

In an on-line network system, the CPU processing rate may vary with the number of customers receiving the CPU processing service. If the number of customers receiving the service simultaneously increases beyond the CPU capacity, the processing rate will decrease rapidly due to the increase of page faults and swappings, interferences and etc. In order to enhance the system efficiency, the number of running terminals should be then controlled appropriately. As we shall see, our analysis in this paper can be applied to get the optimal number of running terminals in system. By optimal, we mean the one which minimizes the mean total system time of customers.

In this paper, we present a model for on-line network systems and then analyse it approximately using queueing theory. Our results are so simple that they provide the optimal number of terminals in system with small computational efforts. In order to analyse the model approximately we divide the whole system into a computer subsystem and some queueing subsystems. The computer subsystem describes state transitions of terminals while the queueing subsystems describe queueing aspects of customers at terminals. We analyse the computer subsystem by formulating it as a closed queueing network and replacing one of its nodes with a flow equivalent exponential node. The steady-state probabilities for the computer subsystem allow us to estimate approximately the turn-around-time distribution of jobs which customers input at terminals. Then the mean total system time of customers can be approximately obtained by applying an approximation method for $M/G/m$ queues.

In the next section, we describe the model for on-line network systems in detail. An approximate analysis for the model is then given in Section 3. In Section 4 some numerical results are present to show the effect of terminal control. In Section 5 some discussions about the accuracy of the approximations are given. In Section 6 another approximate analysis is provided for the cases of heavy traffic and overload traffic. As an application, in Section 7, we propose a terminal control policy based on our analyses.

2. Model Description

The on-line network system to be considered in this paper is shown in Figure 2.1. The system consists of a single CPU and a large number of terminals located in M different service stations. Although it might be restrictive,[†] we assume that each station has the same number of terminals, say m terminals. At each station, customers arrive via a Poisson process with a common arrival rate λ and each customer receives a CPU service through one of the m terminals. The service discipline associated with each station is first-come-first-served.

The service requirements brought by customers to the CPU are exponentially distributed with mean $1/\mu_2$. A customer needs a random time to input his service requirement through a terminal before receiving the CPU processing. After waiting that the CPU processing is completed, he will get an output through the same terminal that takes a random duration. Thus the service process for a customer is composed of the three stages: (1) input service, (2) CPU processing and (3) output service. A customer cannot initiate the input stage until the previous customer, who uses the same terminal, completes all the service process. We assume that the input and the output service times for a customer

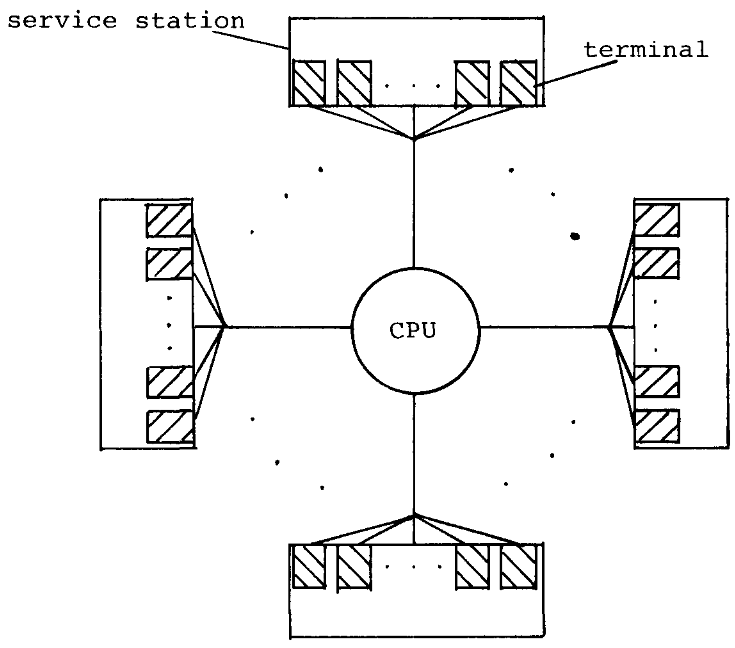

Figure 2.1

[†]An extended model for on-line network systems with different numbers of terminals and arrival rates at each station will be studied in a forthcoming paper.

are exponentially distributed random variables with means $1/\mu_1$ and $1/\mu_3$ respectively, and the service times spent in the three stages are all mutually independent.

The CPU processes service requirements from all terminals in a time-sharing fashion. The CPU processing time for a customer is therefore dependent on the number of customers in the CPU processing stage. Moreover, the CPU processing rate may vary with the number of terminals receiving the CPU processing simultaneously. Let $s(n)$ denote the CPU processing rate when there are n customers in the CPU processing service stage. At that time, each customer receives service with rate $s(n)/n$. We assume that $s(n)$ is a concave function as shown in Figure 2.2. The function $s(n)$ increases as n increases until some n_0. When n exceeds n_0, the CPU reaches an overloaded state and $s(n)$ begins to decrease. Some examples of such service rate functions can be found in [3].

In the real world, the number M of service stations in such on-line network systems as described above is in general quite large. Therefore we deal with the case that has a large M.

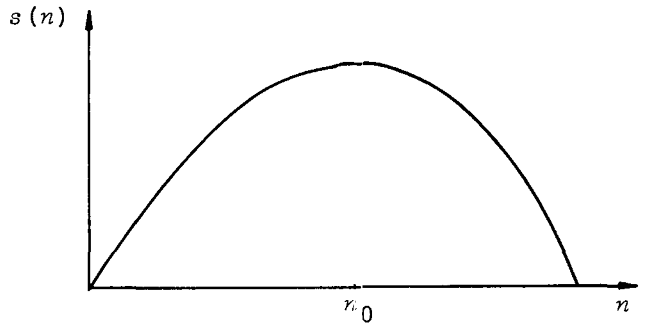

Figure 2.2

3. Analysis

We shall analyse the model described in the previous section under equilibrium condition. Since the model is quite large, however, it is difficult, or perhaps impossible, to get an exact solution. If we attempt to formulate the entire system as a single stochastic process, the state of the process has to include a great quantity of information such as the queue sizes of all stations and the stages of service for all customers receiving the service. Hence not only the number of states but also the dimension of the state space becomes enormous.

In this section, an approximate analysis for the on-line network system is given. The approximation is mainly due to considering the computer subsystem and the queueing subsystems at each station separately. The computer subsystem consists of the CPU and all the terminals. The queueing subsystem at a station consists of customers and the terminals which are referred to as the servers of that station. First we derive the steady-state distribution for the computer subsystem and then use it to analyse the queueing subsystems.

3.1. Analysis of the computer subsystem

The computer subsystem can be represented as a closed queueing network with four nodes as shown in Figure 3.1. Here the terminals at stations play the role of customers in ordinary queueing networks. Each node in the closed network corresponds to one of the four possible states of terminals. We say a terminal is in node i (i=1, 2, 3) if the service through the terminal is in the ith stage, (recall the definition of stages), and it is in node 4 if it is in idle state.

A terminal moves from node 1 to node 2 and then to node 3 in an obvious way. When the service through a terminal is completed, the terminal moves from node 3 to node 1 or to node 4 according

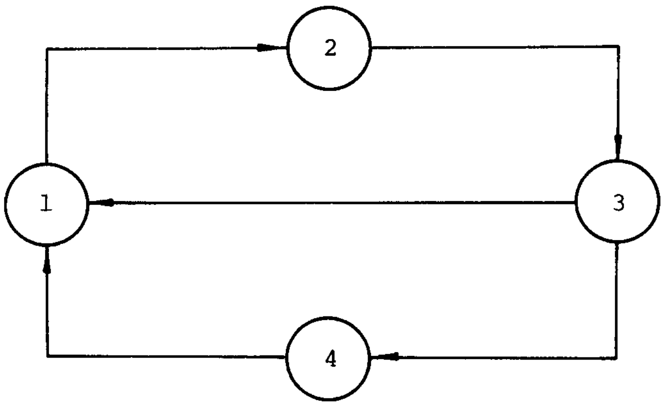

Figure 3.1

to whether or not there is any customer waiting for the service at the corresponding station. If an arrival at a station finds an idle terminal then the terminal moves from node 4 to node 1. nodes 1, 3 and 4 can be considered as infinite-server nodes and the discipline of the service at node 2 is processor sharing. The service times at nodes 1 and 3 are exponentially distributed with means $1/\mu_1$ and $1/\mu_3$. The CPU service requirement of each terminal at node 2 is also exponentially distributed and the mean number of service completions per unit time at node 2 is $\mu_2 s(n_2)$ when there are n_2 terminals there.

The sojourn time of a terminal at node 4 and the transition probabilities that a terminal leaving node 3 moves next to node 1 or to node 4 depend on the state of the service station where the terminal is located. To cope with the difficult situation, we transform the model in Figure 3.1 to the one in Figure 3.2 by allowing zero service time at node 4. The transformed model in Figure 3.2 is however still difficult to analyse, because the service time distribution at node 4 is not simple. We shall then further make an approximation by considering the service time distribution as being exponential. By this approximation, the state of the computer subsystem becomes independent of the state of the queueing subsystems and the computer subsystem becomes possible to be analysed by itself.

The new exponential node 4 should keep the flow rate unchanged. Note that the total arrival rate in the network service

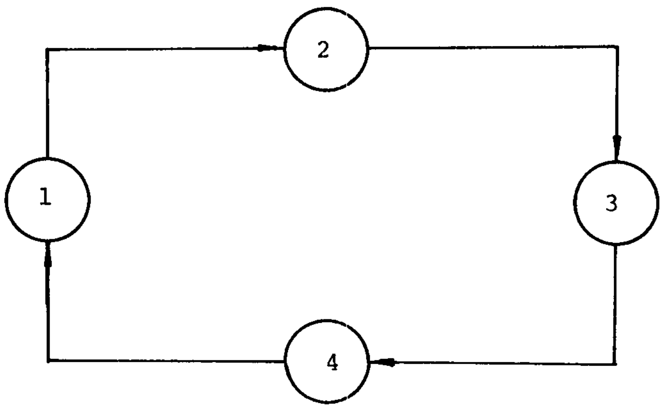

Figure 3.2

system, which is also the flow rate at node 4 in Figure 3.2 under equilibrium condition, is λM. Hence we let the mean number of total service completions per unit time at node 4 be λM.

Thus far, we have obtained a simple approximate model for the computer subsystem. The model is a closed queueing network with a product form solution studied in [1]. Let $\boldsymbol{n}=(n_1, n_2, n_3, n_4)$ be the state of the computer subsystem, where n_j represents the number of terminals in node j ($j=1,2,3,4$). It should be kept in mind that

(3.1) $n_1 + n_2 + n_3 + n_4 = Mm.$

The approximate solution for the steady-state probability $\pi(\boldsymbol{n})$ is given by

$$\text{(3.2)} \qquad \pi(\boldsymbol{n}) \approx \frac{C}{n_1!\, \mu_1^{n_1} \left(\prod_{r=1}^{n_2} s(r) \right) \mu_2^{n_2}\, n_3!\, \mu_3^{n_3} (\lambda M)^{n_4}}, \quad \boldsymbol{n} \in \Phi,$$

where $\Phi = \{\boldsymbol{n} \mid n_j \geqq 0, j=1,2,3,4, \; n_1+n_2+n_3+n_4=Mm\}$ and C is a normalizing constant chosen so that the sum of probabilities is unity, i.e.,

$$\text{(3.3)} \qquad C = \left[\sum_{\boldsymbol{n}\in\Phi} \frac{1}{n_1!\, \mu_1^{n_1} \left(\prod_{r=1}^{n_2} s(r) \right) \mu_2^{n_2}\, n_3!\, \mu_3^{n_3} (\lambda M)^{n_4}} \right]^{-1}.$$

Here, for convention, we define $\prod_{r=1}^{0} s(r)=1$. Accuracy of this approximation formula will be discussed in Section 5.

3.2. Analysis of the queueing subsystems at service stations

We have assumed that at each service station customers arrive according to a Poisson process with rate λ and receive service through m terminals in the first-come-first-served order. From the assumption about the CPU processing rate, the service time depends on the states of all stations. If, however, the number of stations is assumed to be large, one may ignore the dependence. We then consider each station in isolation, where the service time has a state-independent distribution calculable from the steady-state distribution for the computer subsystem. That is, the queueing subsystem at each station is approximately considered as an ordinary *M/G/m* system.

The exact analysis for *M/G/m* systems is known to be difficult and many approximation methods have been developed. We shall apply one of those methods to obtain an approximate solution for the mean waiting time of the queueing subsystem. The quantities necessary associated with the service time can be calculated from (3.2) and (3.3). Here, we use a simple result in [2]. After a little algebra, the approximation formula of [2] for the mean waiting time of *M/G/m* systems is transformed to

$$W(M/G/m) \approx \frac{1+c_s^2}{2}\left[\frac{1-c_s^2}{1+(1-\rho)(m-1)\frac{\sqrt{4+5m}-2}{16m\rho}} + c_s^2\right]^{-1} W(M/M/m), \tag{3.4}$$

where, c_s and ρ are the coefficient of variation of service time and the utilization factor of system respectively. Let ES and ES^2 be the first and the second moments of service time. We then have

$$c_s^2 = \frac{ES^2-(ES)^2}{(ES)^2}, \tag{3.5}$$

and

$$\rho = \frac{\lambda ES}{m}. \tag{3.6}$$

The mean waiting time for *M/M/m* queues is known as

$$W(M/M/m) = \frac{ES\, m^{m-1}\rho^m}{m!(1-\rho)^2}\left[\sum_{n=0}^{m-1}\frac{\rho^n}{n!}m^n + \frac{m^m\rho^m}{m!(1-\rho)}\right]^{-1}. \tag{3.7}$$

In order to calculate the approximate value of the mean waiting time from (3.4), we need the first two moments of the service time distribution. Recall that the three parts of the service time corresponding to the three service stages (1), (2) and (3) are independent of each other. Hence if we denote the CPU processing time by S_2, one has

$$P(S_2 \leq x) = \sum_{n=1}^{Mm}\frac{\pi_2(n)}{1-\pi_2(0)}\cdot(1-\exp(-\frac{\mu_2 s(n)}{n}x)),$$

where $\pi_2(n)$ is the steady-state distribution of the total number of customers in the CPU processing stage. Thus the first two moments of the service time can be derived as

(3.8) $$ES = \frac{1}{\mu_1} + \frac{1}{\mu_2} \sum_{n=1}^{Mm} \frac{\pi_2(n)}{1-\pi_2(0)} \frac{n}{s(n)} + \frac{1}{\mu_3},$$

and

(3.9) $$ES^2 = \frac{2}{\mu_1^2} + \frac{2}{\mu_2^2} \sum_{n=1}^{Mm} \frac{\pi_2(n)}{1-\pi_2(0)} \left(\frac{n}{s(n)}\right)^2 + \frac{2}{\mu_3^2}$$
$$+\left(\frac{2}{\mu_1\mu_2} + \frac{2}{\mu_2\mu_3}\right) \sum_{n=1}^{Mm} \frac{\pi_2(n)}{1-\pi_2(0)} \frac{n}{s(n)} + \frac{2}{\mu_1\mu_3}.$$

We need to calculate $\pi_2(n)$, which is the marginal distribution of state n for the computer subsystem. From (3.2) and (3.3),

(3.10) $$\pi_2(n) \approx \frac{\sum_{\boldsymbol{n}\in\Phi(n)} [\mu_1^{n_1} \mu_2^{n} \mu_3^{n_3} (\lambda M)^{n_4} n_1!\,(\prod_{r=1}^{n} s(r))\,n_3!]^{-1}}{\sum_{\boldsymbol{n}\in\Phi} [\mu_1^{n_1} \mu_2^{n_2} \mu_3^{n_3} (\lambda M)^{n_4} n_1!\,(\prod_{r=1}^{n_2} s(r))\,n_3!]^{-1}},$$

where, $\Phi(n)=\{\boldsymbol{n} \mid n_j \geqq 0, j=1,2,3,4,\ n_2=n,\ n_1+n_3+n_4=Mm-n\}$. The right hand side of (3.10) can be simplified in the following way:

$$\text{numerator} = \frac{1}{\mu_2^n \prod_{r=1}^{n} s(r)} \sum_{n_4=0}^{Mm-n} \frac{1}{(\lambda M)^{n_4}} \left[\sum_{n_1+n_3=Mm-n-n_4} \frac{1}{\mu_1^{n_1} \mu_3^{n_3} n_1!\, n_3!}\right]$$

$$= \frac{1}{\mu_2^n \prod_{r=1}^{n} s(r)} \sum_{n_4=0}^{Mm-n} \frac{1}{(\lambda M)^{n_4} (Mm-n-n_4)!} \left(\frac{1}{\mu_1} + \frac{1}{\mu_3}\right)^{Mm-n-n_4}$$

$$= \frac{1}{\mu_2^n (\prod_{r=1}^{n} s(r))(\lambda M)^{Mm-n}} \sum_{k=0}^{Mm-n} \frac{1}{k!}\left[\lambda M\left(\frac{1}{\mu_1} + \frac{1}{\mu_3}\right)\right]^k;$$

and similarly,

$$\text{denominator} = \sum_{n_2=0}^{Mm} \left\{\frac{1}{\mu_2^{n_2} (\prod_{r=1}^{n_2} s(r))(\lambda M)^{Mm-n_2}} \cdot \sum_{k=0}^{Mm-n_2} \frac{1}{k!}\left[\lambda M\left(\frac{1}{\mu_1} + \frac{1}{\mu_3}\right)\right]^k\right\}.$$

Combining these, one finally has

$$(3.11)\qquad \pi_2(n) \approx \frac{\dfrac{(\lambda M)^n}{\mu_2^n\left(\prod_{r=1}^{n} s(r)\right)} \sum_{k=0}^{Mm-n} \dfrac{1}{k!}\left[\lambda M\left(\dfrac{1}{\mu_1}+\dfrac{1}{\mu_3}\right)\right]^k}{\sum_{n_2=0}^{Mm}\left\{\dfrac{(\lambda M)^{n_2}}{\mu_2^{n_2}\left(\prod_{r=1}^{n_2} s(r)\right)} \sum_{k=0}^{Mm-n_2} \dfrac{1}{k!}\left[\lambda M\left(\dfrac{1}{\mu_1}+\dfrac{1}{\mu_3}\right)\right]^k\right\}}\,.$$

Using (3.4) through (3.9) and (3.11), we obtain the mean waiting time W of the queueing subsystem at each service station.

3.3. Some quantities for system performance measures

Using the approximate results derived in Sections 3.1 and 3.2, some quantities for system performance measures can be obtained. Let T be the mean total time spent in the on-line network system by customers. We then have

$$(3.12)\qquad T = W + ES.$$

Suppose that M, $s(n)$ and μ_j (j=1, 2, 3) are fixed. Let $\rho_m(\lambda)$ and $T_m(\lambda)$ be respectively the utilization factor of the terminals and the mean total system time of customers as functions of λ for given m. Define a_m as the critical value of λ at which the system becomes unsteady when there are m terminals at each station. Namely, $\lim_{\lambda\to a_m} \rho_m(\lambda)=1$ and consequently $\lim_{\lambda\to a_m} T_m(\lambda)=\infty$. a_m can be calculated by substituting 1 for the left hand side of (3.6).

Our approximate analysis for the on-line network system has been done under equilibrium condition. We note, however, that even if the queueing subsystems get overloaded, i.e. $\rho \geqq 1$, the computer subsystem may still be steady. In this case, the mean total system time is infinite and therefore the discussion about it becomes meaningless, but the CPU throughput, denoted by R, is of interest to us. We have the following relation regardless of whether $\rho<1$ or $\rho \geqq 1$:

$$(3.13)\qquad R = \sum_{n=0}^{Mm} \pi_2(n)\,\mu_2 s(n).$$

The steady-state distribution $\pi_2(n)$ in the case of $\rho \geqq 1$ will be derived in Section 6.

4. Numerical Examples

Numerical results for some examples are shown in Figures 4.1 through 4.4, from which the mean total system times of the cases m = 3, 4, 5 and 6 can be compared. In the example illustrated in Figure 4.1, we set $\mu_1 = 2$, $\mu_2 = 1$, $\mu_3 = 6$, $M = 100$ and $s(n)$ is a quadratic curve. We see that $T_m(\lambda)$ is an increasing function and becomes infinite as λ approaches to a_m. The larger m, the more rapidly $T_m(\lambda)$ near a_m increases. For small λ the difference in the values of $T_m(\lambda)$ with respect to m, $m = 3, 4, 5, 6$, is not significant while for large λ the difference becomes notable. Here we take the mean total system time T as the performance metric. We refer to the number of terminals as the optimal at which the smallest mean total system time is achieved. In this example, we explicitly observe the difference among the functions $T_m(\lambda)$ for various m so that terminal control should be made where $m = 4$ is optimal for large λ. Figures 4.2 and 4.3 illustrate two other examples in which the same parameter values as those in Figure 4.1 are used except for the function $s(n)$. In these two examples, the optimal numbers of terminals are 5 and 6 respectively. Compared with Figure 4.2, we observe that in Figure 4.4, a_6 becomes smaller and a_3 becomes larger relatively. If we make μ_2 smaller, with μ_1 and μ_3 fixed, ES_2 will take a larger portion in ES, so that the affect of the CPU processing rate function $s(n)$ will become significant.

With M, m and $s(n)$ fixed, one sees from (3.6), (3.8) and (3.11) that the utilization factor ρ of the queueing subsystems is invariant as long as the relative values of λ and μ_i (i=1, 2, 3), i.e., λ / μ_i and μ_i / μ_j (i, j=1, 2, 3), are not changed. Note that a_m represents the arrival rate corresponding to $\rho = 1$ when the number of terminals is m. Thus the relative positions of $\{a_m\}$ are invariant provided that M, $s(n)$ and the relative values of $\{\mu_j\}$ are not changed. Therefore, one may guess that if two on-line network systems have the same M, $s(n)$ and relative values of $\{\mu_j\}$, they would have similar curves of $T_m(\lambda)$ and probably the same optimal number of terminals. For any on-line network system, it is clear that the optimal m is not smaller than n_0 divided by M where n_0 is the number which maximizes $s(n)$. The difference between the values of the optimal m and n_0/M greatly

$$\mu_1=2,\ \mu_2=1,\ \mu_3=6,\ M=100,\ s(n)=\begin{cases}100-\dfrac{(n-300)^2}{900}, & 0\leqslant n\leqslant 600,\\ 0, & \text{otherwise.}\end{cases}$$

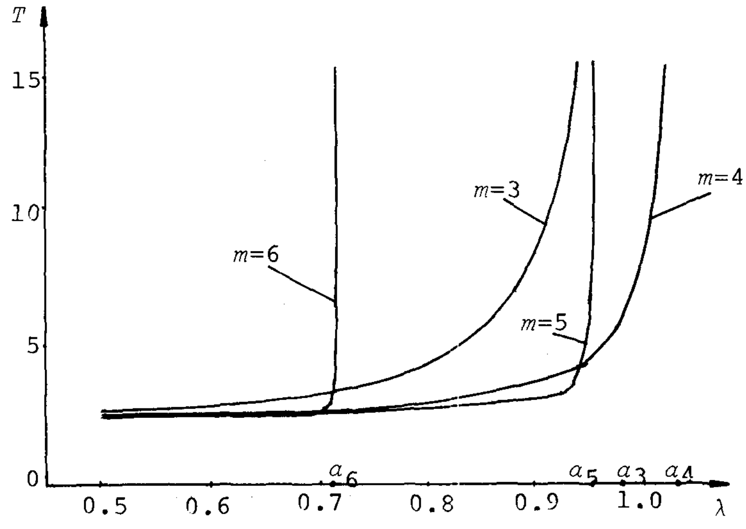

Figure 4.1

$$\mu_1=2,\ \mu_2=1,\ \mu_3=6,\ M=100,\ s(n)=\begin{cases}100-\dfrac{(n-400)^2}{1600}, & 0\leqslant n\leqslant 400,\\ 100-\dfrac{(n-400)^2}{400}, & 400< n\leqslant 600,\\ 0, & \text{otherwise.}\end{cases}$$

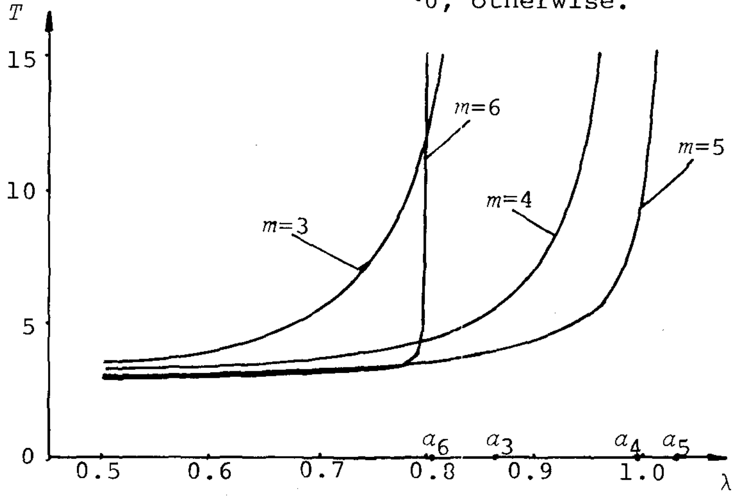

Figure 4.2

$$\mu_1=2,\ \mu_2=1,\ \mu_3=6,\ M=100,\ s(n)=\begin{cases}100-\dfrac{(n-500)^2}{2500}, & 0\leqslant n\leqslant 500,\\[4pt] 100-\dfrac{(n-500)^2}{100}, & 500<n\leqslant 600,\\[4pt] 0, \text{ otherwise.}\end{cases}$$

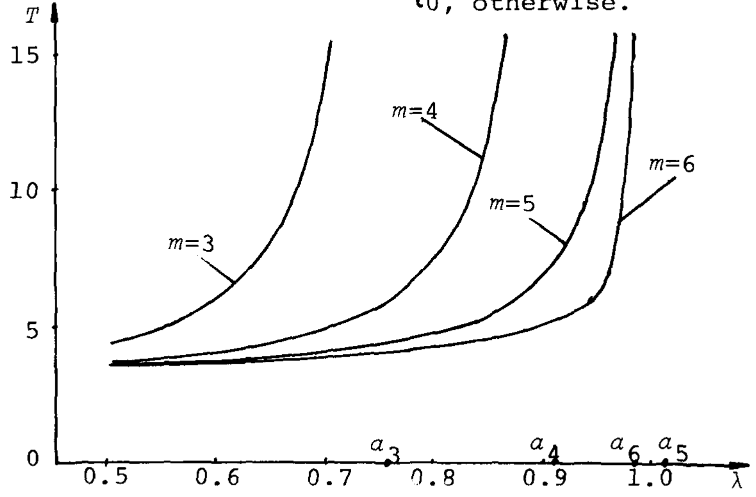

Figure 4.3

$$\mu_1=2,\ \mu_2=0.5,\ \mu_3=6,\ M=100,\ s(n)=\begin{cases}100-\dfrac{(n-400)^2}{1600}, & 0\leqslant n\leqslant 400,\\[4pt] 100-\dfrac{(n-400)^2}{400}, & 400<n\leqslant 600,\\[4pt] 0, \text{ otherwise.}\end{cases}$$

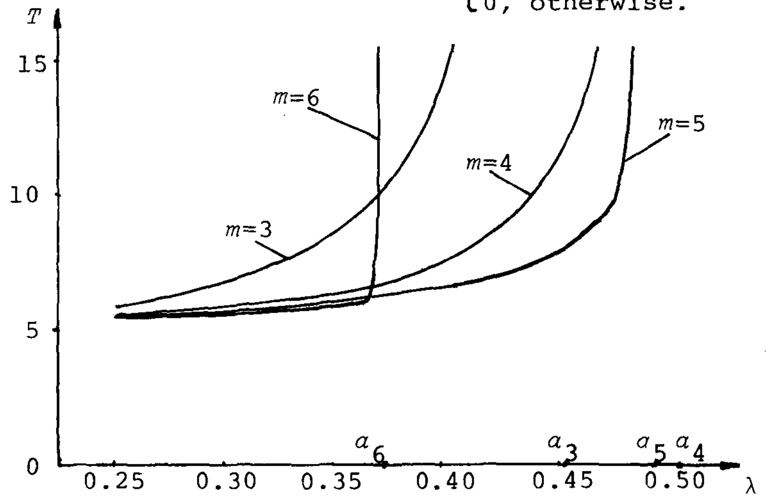

Figure 4.4

depends on the slope of $s(n)$ after n_0. In the examples illustrated in Figures 4.1 through 4.4, in which the CPU processing rate functions $s(n)$ decrease rapidly after n_0, the differences are all equal to 1.

5. Approximation Accuracy

We have introduced three steps of approximation in Section 3. First, we replaced node 4 in Figure 3.2 with an exponential node. By this approximation, the state of the queueing subsystems was ignored. The second approximation is due to formulating each queueing subsystem as an *M/G/m* queue. Moreover, we applied an approximation formula with respect to *M/G/m* queues. We suspect that the second approximation is not significant compared with the other two approximations when the number of stations, *M*, is large. The accuracy of the third approximation is dependent on the formula to be used. In the remainder of this section, we only consider the accuracy of the approximation formula (3.2) due to the first approximation. We cannot calculate the error of (3.2) directly, because the exact value of $\pi(\boldsymbol{n})$ is difficult to obtain. Also the enormous size of the system makes a simulation quite difficult. Here we check the accuracy in two different ways which do not show the accuracy directly but give us information on the accuracy to a certain extent.

The first way is to check to what degree the approximate solution satisfies the node balance equations which equate the flow into each node to the flow out of that node. The node balance equations for the computer subsystem (see Figure 3.1) are established as follows:

(5.1) $$P(n_1,n_2,n_3,n_4)\,\mu_1 n_1 = P(n_1-1,n_2,n_3,n_4+1)\,\lambda\,M\,P\{A \mid n_4+1\} + P(n_1-1,n_2,n_3+1,n_4)\,\mu_3(n_3+1)(1-P\{D \mid n_4\})$$
$(\boldsymbol{n}\in\Phi$ and $n_1 \geqq 1)$,

(5.2) $$P(n_1,n_2,n_3,n_4)\,\mu_2 s(n_2) = P(n_1+1,n_2-1,n_3,n_4)\,\mu_1(n_1+1)$$
$(\boldsymbol{n}\in\Phi$ and $n_2 \geqq 1)$,

(5.3) $$P(n_1,n_2,n_3,n_4)\,\mu_3 n_3 = P(n_1,n_2+1,n_3-1,n_4)\,\mu_2 s(n_2+1)$$
$(\boldsymbol{n}\in\Phi$ and $n_3 \geqq 1)$,

(5.4) $$P(n_1,n_2,n_3,n_4)\,\lambda\,M\,P(A \mid n_4) = P(n_1,n_2,n_3+1,n_4-1) \times \mu_3(n_3+1)P\{D \mid n_4-1\}$$
$(\boldsymbol{n}\in\Phi$ and $n_4 \geqq 1)$,

where,

$P\{A \mid n_4\}$ = P\{an arrival in the on-line network system will find an idle terminal | currently there are n_4 idle terminals in total\},

$P\{D \mid n_4\}$ = P\{a departure from the on-line network system will leave the corresponding terminal idle | currently there are n_4 idle terminals in total\}.

From (5.4), we can rewrite (5.1) as

(5.5) $$P(n_1, n_2, n_3, n_4)\,\mu_1 n_1 = P(n_1-1, n_2, n_3+1, n_4)\,\mu_3 (n_3+1)$$
$$(n \in \Phi \text{ and } n_1 \geqq 1).$$

It can be readily checked that the approximate solution shown in (3.2) satisfies (5.2), (5.3) and (5.5). Unfortunately, (5.4) is not satisfied because $P\{A \mid n_4\} \neq P\{D \mid n_4-1\}$ in general. However we suspect that $P\{A \mid n_4\}$ is rather close to $P\{D \mid n_4-1\}$ if n_4 is not extremely small. Hence the distribution shown in (3.2) should have a good accuracy.

The accuracy can also be checked by a numerical means. Note that, when the on-line network system is in equilibrium state, the total arrival rate and the total departure rate must be consistent. In other words, if the total departure rate calculated by the approximation formula (3.2) is very different from the total arrival rate then we have to say that the accuracy of the approximation is poor. Now let us calculate the total departure rate and compare it with the total arrival rate. Note that the total departure rate from the on-line network system is just the same as the CPU throughput R. Table 5.1 shows the utilization factors ρ and the CPU throughputs R calculated approximately by (3.6), (3.8), (3.11) and (3.13) for the cases illustrated in Figure 4.2. (Note that (3.6), (3.8) and (3.13) are not approximation formulas, and (3.11) was derived from (3.2).) We only observe the values in Table 5.1 corresponding to $\rho < 1$, i.e., the values not bracketed. λ is the arrival rate at each station and in this case the number of stations is M = 100, so the total arrival rate in the entire system is 100 λ. From Table 5.1, we see that when $\rho < 1$, R is roughly consistent with 100 λ, which again means that the accuracy of the approximation should be sufficiently reliable. We also observe from Table 5.1 that the accuracy is not yet so satisfactory when ρ is near to 1 (i.e., when the value of n_4 is likely small).

Table 5.1

λ	$m=3$	$m=4$	$m=5$	$m=6$
0.7	ρ =0.76 R=70.00	0.57 70.00	0.46 70.00	0.38 70.00
0.8	0.91 79.18	0.69 80.00	0.55 80.00	[1.43 78.90]
0.9	[1.05 82.92]	0.84 89.92	0.67 90.00	[78.33 1.44]
1.0	[1.17 83.81]	1.00 95.74	0.91 98.60	[87.08 1.44]

6. Heavy and Overload Cases

From Section 5, it seems that the accuracy of the approximate solution in Section 3.1 is not sufficient when ρ is near to 1. Here we give another analysis for the case of heavy traffic ($\rho \approx 1$ but $\rho < 1$) which can also be applied to the case of overload traffic ($\rho \geqq 1$). Note that in these cases it scarcely occurs that there is any idle terminal. Thus we omit node 4 from Figure 3.1 for the computer subsystem as shown in Figure 6.1. The service at each node has the same rate and discipline as in Section 3.1. Here we take the state of the computer subsystem as $\boldsymbol{n} = (n_1, n_2, n_3)$, where n_1, n_2 and n_3 represent the same variables as in Section 3.1 with

(6.1) $n_1 + n_2 + n_3 = Mm.$

Again this queueing network has a product form solution and the distribution of the number of terminals in node 2 is given by

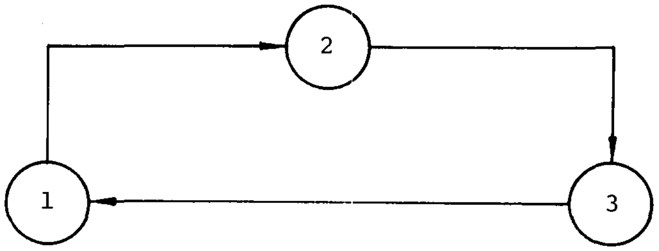

Figure 6.1

$$(6.2) \qquad \pi_2(n) = \frac{\dfrac{1}{\mu_2^n\left(\prod_{r=1}^{n} s(r)\right)(Mm-n)!}\left(\dfrac{1}{\mu_1}+\dfrac{1}{\mu_3}\right)^{Mm-n}}{\sum_{n_2=0}^{Mm}\dfrac{1}{\mu_2^{n_2}\left(\prod_{r=1}^{n_2} s(r)\right)(Mm-n_2)!}\left(\dfrac{1}{\mu_1}+\dfrac{1}{\mu_3}\right)^{Mm-n_2}} .$$

Note that in the above derivation no approximation is made, i.e., (6.2) is just an exact result under the condition (6.1).

When $\rho \geqq 1$, the mean total time is of no meaning, but the quantity of CPU throughput is meaningful. Table 6.1 shows the throughputs R of the same cases as in Table 5.1 calculated by using (6.2). Here, R is independent of λ. We see that in this example $m = 4$ is optimal and it will lead to a serious situation to let 6 terminals be running when λ is so large that $\rho \approx 1$ or $\rho \geqq 1$ for each of the four choices of m.

Table 6.1

m	3	4	5	6
R	84.67	97.33	96.66	1.41

7. Terminal Control Policy

We have seen from the analyses for the on-line network system that terminal control is necessary and effective. In this section, we propose a terminal control policy as follows: Let μ_i (i=1, 2, 3), M and $s(n)$ be fixed. When the arrival rate λ is given, calculate the utilization factor ρ first for each of the candidates for the number of terminals using (3.6), (3.8) and (3.11). If ρ is not very large, say, smaller than 0.95 for some candidates, then calculate the mean total system time T for these candidates using the results in Section 3 and choose the one with the smallest mean total system time as the optimal number of terminals. If $\rho \approx 1$ or $\rho \geqq 1$ for all the candidates, then calculate the CPU throughput R for each of the candidates using (3.13) and (6.2) and choose the one with the largest CPU throughput.

In some situations, the arrival rate may change from time to time. To keep the system to be optimal all the time, terminal control should be made by running more terminals or stopping some running terminals according to the change of λ . The sooner the decision is made, the better. All the quantities needed to the decision are easy to calculate by computer. Hence the approximate method of this paper hopefully provides a means of terminal control.

Acknowledgements

The authors should be grateful to the refrees for helpful comments and suggestions. They also thank Dr. M. Kijima for generous discussions and carefully reading the manuscript of this paper.

References

[1] Baskett, F., Chandy, K.M., Muntz, R.R. and Palacios, F.G.: Open, Closed and Mixed Networks of Queues with Different Classes of Customers. *Journal of the Association for Computing Machinery*, Vol.22, No.2 (1975), 248-260.

[2] Kimura, T.: A Two-Moment Approximation for the Mean Waiting Time in the *GI/G/s* Queue. *Research Reports on Information Sciences*, B-154 (1984), Tokyo Institute of Technology.

[3] Kleinrock, L.: *Queueing Systems, Volume II : Computer Applications*. John Wiley & Sons, 1975.

Huanxu PAN and Hidenori MORIMURA:
Department of Information Sciences, Tokyo Institute of Technology, Oh-okayama, Meguro-ku, Tokyo 152, Japan.