A SIMULATION TOOL TO EVALUATE THE PERFORMANCE OF FINITE-SOURCE QUEUEING MODELS*

B. Almási and J. Sztrik (Debrecen, Hungary)

This paper deals with a software tool to evaluate the main characteristics of a nonhomogeneous finite-source queueing model to describe the performance of a multi-terminal system subject to random breakdowns under FIFO, priority processor sharing, and polling service disciplines. The model studied here is actually a closed queueing network with three nonindependent service stations (CPU, terminals, and repairman), and a finite number of customers (jobs), which have different service rates at the service stations. The aim of this paper is to introduce the FQM (finite-source queueing model) program package, which was developed at the Institute of Mathematics and Informatics of Lajos Kossuth University in Debrecen, Hungary, and to investigate the performance of the above-mentioned finite-source queueing models. At the end we give a sample result to illustrate the problem in question.

1. Introduction

In recent years, finite-source queueing models in different forms have been efficiently used, for example, for a mathematical description of information systems (see [7]). Several works have been devoted to nonreliable queueing models. This paper deals with a model of an information system consisting of n terminals connected with a service provider (CPU or server) and a repairman to repair the breakdowns of the CPU and terminals. The thinking times and the request processing times of the user at terminal i are random variables with distribution function depending on index i. Different models can be constructed by using different service disciplines at the CPU (e.g., FIFO, priority processor sharing, polling). Let us suppose that the CPU is subject to random breakdowns stopping the entire system, and giving a special repair to the operative. The busy terminals are also subject to random breakdowns not affecting the system's operation but stopping the work at the terminal, and thus giving duties to the repairman. The failure-free operation times and repair times of busy terminals are assumed to be random variables with mean depending on the index of the terminal. The breakdowns are serviced by a repairman according to the FIFO discipline among terminals and providing preemptive priority to the failure of the CPU. We assume that each user generates only one job at a time, and he sleeps until its job is serviced, that is, the terminal is inactive while waiting at the CPU, and it cannot break down. Assuming that all random variables are exponentially distributed and independent of each other, the system's behavior can be described by a Markov chain, having a complex and rather huge state space. We show an efficient computational method and introduce software tools that can be used to calculate the most important steady-state characteristics of the system, such as utilization of the CPU, average response times, and average length of the CPU's and repairman's queue.

On the one hand, this paper is a generalization of the nonhomogeneous model discussed in [4, 5] (which allowed only the breakdown of the CPU); on the other hand, it further generalizes the homogeneous model treated in [6] (which allowed both terminal and CPU failures).

2. Model Formulation

To deal with the problem we have to introduce the following random variables:

\[
X(t) = \begin{cases} 
1, & \text{if the operating system fails at time } t, \\
0, & \text{otherwise,}
\end{cases}
\]

\[
Y(t) = \text{the number of failed terminals at time } t,
\]

---

*Supported by the Hungarian National Foundation for Scientific Research (grant Nos. OTKA T014974/95 and T016933/95).
\( Y(t) = \) the failed terminals' indices at time \( t \) in order of their failure, or 0 if \( Y(t) = 0 \),

\( Z(t) = \) the number of jobs residing at the CPU at time \( t \),

\( ZI(t) = \) the indices of these jobs.

Depending on the service discipline, the random variable \( ZI(t) \) gives the order of service by the CPU also. It can easily be seen that the stochastic process \( M(t) = (X(t), Y(t), YI(t), Z(t), ZI(t)) \) is a Markov chain having a rather complex and large state space. To get its steady-state probabilities an efficient recursive computational method has been introduced and used for the different service rules mentioned earlier (cf. [1, 2, 6]). As can be seen in Table 1, our method needs only 10\% of the traditional algorithm's memory requirement.

| TABLE 1 |
|-----------------|---------|---------|
| Memory requirements of the traditional algorithm | \( n = 4 \) | \( n = 5 \) | \( n = 6 \) |
| Memory requirements of our method | 138 KB | 4846 KB | 238474 KB |

Knowing the steady-state probabilities and denoting by \( P(q, k, s) \) the steady-state probability that the operating system is in state \( q \), \( k \) terminals fail, and \( s \) jobs are at the CPU, the main performance measures can be obtained with a simple summation as follows (the formulas can be found in [1, 3] also).

(i) Mean number of jobs residing at the CPU:

\[
\bar{n}_j = \sum_{i=0}^{1} \sum_{k=0}^{n} \sum_{s=0}^{n-k} sp(i, k, s).
\]

(ii) Mean number of good terminals:

\[
\bar{n}_g = n - \sum_{i=0}^{1} \sum_{k=0}^{n} \sum_{s=0}^{n-k} kp(i, k, s).
\]

(iii) Average number of busy terminals:

\[
\bar{n}_b = \sum_{k=0}^{n} \sum_{s=0}^{n} (n - k - s)p(0, k, s).
\]

(iv) Utilization of the repairman:

\[
U_r = \sum_{k=0}^{n} \sum_{s=0}^{n-k} p(1, k, s) + \sum_{k=1}^{n} \sum_{s=0}^{n-k} p(0, k, s).
\]

(v) Utilization of the CPU:

\[
U_{CPU} = \sum_{k=0}^{n-1} \sum_{s=1}^{n-k} p(0, k, s).
\]

(vi) Utilization of terminal \( i, i = 1, \ldots, n \):

\[
U_i = \sum_{k=0}^{n} \sum_{s=0}^{n-k} \sum_{r=1}^{k} \sum_{v=1}^{k} \sum_{j_1, \ldots, j_s} (1 - \delta(i, i_r) - \delta(i, j_v))p(0; k; i_1, \ldots, i_k; s; j_1, \ldots, j_s).
\]

(vii) Expected response time of jobs for terminal \( i \):

\[
T_i = \frac{Q_i}{\lambda_i U_i}.
\]