Acta Cybernetica, Vol. 10, Nr. 4, Szeged, 1992

A Queuing Model for a Processor-Shared Multi-Terminal System Subject to Breakdowns

B. Almási*

Abstract

This paper deals with a non-homogeneous finite-source queuing model to describe the performance of a multi-terminal system subject to random breakdowns under PPS (Priority Processor-Sharing) service discipline. It can be viewed as a continuation of paper [1], which discussed a FIFO (First-In, First-Out) serviced queuing model subject to random breakdowns. All random variables are assumed to be independent and exponentially distributed. The system's behaviour can be described by a Markov chain, but the number of states is very large (it is a combinatorically increasing function of the number of terminals). The purpose of this paper is to give a recursive computational approach to solve the steady-state equations and to illustrate the problem in question using some numerical results.

1 The Model

This paper deals with a terminal system consisting of n terminals connected with a Central Processor Unit (CPU). The user at the terminal i thinks for random times and generates jobs to the CPU. The think times are assumed to be exponentially distributed with mean $\frac{1}{\lambda_i}$. The required running times of jobs of terminal i are exponentially distributed random variables with mean $\frac{1}{\mu_i}$ (assuming, that the jobs use the whole capacity of the CPU). The jobs staying at the CPU are serviced in parallel using the PPS scheduling strategy (see [2,3]). Each terminal has a positive weight, denoted by ω_i for terminal $i (i = 1, \ldots, n)$, and if there are $s (1 \leq s \leq n)$ jobs at the CPU from the terminals $j_1, \ldots, j_s$ then the job of the terminal $j_r (r = 1, \ldots, s)$ is serviced at rate

$$W_{j_r}(j_1, \ldots, j_s) = \frac{\omega_{j_r}}{\sum_{i=1}^{s} \omega_{j_i}},$$

that is, the processing intensity is $W_{j_r}(j_1, \ldots, j_s)\mu_{j_r}$ for the job of terminal j_r. Let us suppose that the CPU is subject to random breakdowns stopping the whole system. The failure-free operation times of the CPU are exponentially distributed random variables with mean $\frac{1}{\alpha}$. The restoration times of the CPU are exponentially

*Department of Mathematics, University of Debrecen, Debrecen P.O.Box 12, H-4010, Hungary
Acknowledgement: The author is very grateful to Professor M. Arató and J. Sztrik for their helpful comments. Research is partially supported by Hungarian National Foundation for Scientific Research under grant OTKA 1648/91.

distributed with mean $\frac{1}{\beta}$. The busy terminals are also subject to random breakdowns not affecting the system's operation but stopping the work at the terminal. The failure-free operation times of busy terminals are supposed to be exponentially distributed random variables with mean $\frac{1}{\gamma_i}$ for terminal i. The repair times of terminal i are exponentially distributed random variables with mean $\frac{1}{\tau_i}$. The breakdowns are serviced by a single repairman according to FIFO discipline among terminals and providing preemptive priority to the failure of CPU. We assume that each terminal sleeps while its job is serviced by the CPU, that is, the terminal is inactive while waiting at the CPU, and it cannot break down. All random variables involved here are assumed to be independent of each other.

On the one hand this paper is a generalization of the non-homogeneous PS model discussed in [4] (which allowed only CPU failures), on the other hand it further generalizes the homogeneous model treated in [5] (which allowed both terminal and CPU failures). This paper is the continuation of [1] where the FIFO discipline was discussed (instead of PPS) and we build a new non-homogeneous model and solve the steady-state equations recursively by using a similar computational approach as in [1]. In equilibrium the main performance of the system, such as the mean number of jobs residing at the CPU, the mean number of functional terminals, the expected response time of jobs, and utilizations are obtained. Finally it is investigated - by using some numerical results - how breakdowns affect the performance characteristics and the results of [1] are compared with ours.

2 The Mathematical Model and a Computational Approach

Let us introduce the following random variables:

$$X(t) = \begin{cases} 1, & \text{if the CPU is failed at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

$Y(t) =$ the failed terminals' indices at time t in order of their failure, or 0 if there is no failed terminal,

$Z(t) =$ the indices of jobs staying at the CPU at time t in lexicographically increasing order, or 0 if there is no job at the CPU.

It is easy to see that the process

$$M(t) = (X(t), Y(t), Z(t)),$$

is a multi-dimensional Markov chain with 3 vector-components and with state space

$$S = ((q; i_1, \ldots, i_k; j_1, \ldots, j_s),\ q = 0, 1;\ k = 0, \ldots, n;\ s = 0, \ldots, n-k),$$

where

$(i_1, \ldots, i_k)$ is a permutation of K objects from the numbers $1, \ldots, n$ or 0, if $k = 0$,

$(j_1,\dots,j_s)$ is a combination of s objects from the remaining $n-k$ numbers or 0, if $s=0$.

The event $(q;i_1,\dots,i_k;j_1,\dots,j_s)$ denotes that the operating system is in state $X(t)=q$, there are k failed terminals with indices $i_1,\dots,i_k$, and there are s jobs with indices $j_1,\dots,j_s$ at the CPU $(i_r\neq j_l,\ r=1,\dots,k;\ l=1,\dots,s)$.

The reader can easily verify that the number of states is

$$\dim(S)=2\sum_{k=0}^{n}\sum_{s=0}^{k}\frac{n!}{(n-k)!s!}.$$

Let us denote the steady-state distribution of $(M(t),t\geq 0)$ by

$$p(q;i_1,\dots,i_k;j_1,\dots,j_s)=\lim_{t\to\infty}p(X(t)=q;Y(t)=i_1,\dots,i_k;Z(t)=j_1,\dots,j_s),$$

which exists and is unique (see [6]) because of all the rates are assumed to be positive. For brevity let us introduce the following notation:

$$K_{j_r}(q;i_1,\dots,i_k;j_1,\dots,j_s)=\frac{\omega_{j_r}\mu_{j_r}}{\sum_{i=1}^{s}\omega_{j_i}}p(q;i_1,\dots,i_k;j_1,\dots,j_s),\ r=1,\dots,s.$$

Since we study the steady-state behavior of the Markov chain $M(t)$, following [6], we can start with the statement

$$\begin{aligned}&\text{Average rate of leaving state }(q;i_1,\dots,i_k;j_1,\dots,j_s)=\\&=\text{ Average rate of entering state }(q;i_1,\dots,i_k;j_1,\dots,j_s),\end{aligned}$$

that is, we can build the global balance equations for $p(qi_1,\dots,i_k;j_1,\dots,j_s)$ by using the rules discussed in Section 1:

$$\begin{aligned}(\alpha+\tau_{i_1}+&\sum_{\substack{r\neq i_1,\dots,i_k\\ r\neq j_1,\dots,j_s}}(\lambda_r+\gamma_r))p(0;i_1,\dots,i_k;j_1,\dots,j_s)+\sum_{r=1}^{s}K_{j_r}(0;i_1,\dots,i_k;j_1,\dots,j_s)=\\&=\beta p(1;i_1,\dots,i_k;j_1,\dots,j_s)+\\&+\sum_{\substack{r\neq i_1,\dots,i_k\\ r\neq j_1,\dots,j_s}}(\tau_r p(0;r,i_1,\dots,i_k;j_1,\dots,j_s)+K_r(0;i_1,\dots,i_k;j_1,\dots,r,\dots,j_s))+\\&+\gamma_{i_k}p(0;i_1,\dots,i_{k-1};j_1,\dots,j_s)+\sum_{r=1}^{s}\lambda_{j_r}p(0;i_1,\dots,i_k;j_1,\dots,j_{r-1},j_{r+1},\dots,j_s),\end{aligned}\tag{1}$$

for all $i_1,\dots,i_k;j_1,\dots,j_s;k=0,\dots,n;s=0,\dots,n-k,$

$$\beta p(1;i_1,\dots,i_k;j_1,\dots,j_s)=\alpha p(0;i_1,\dots,i_k;j_1,\dots,j_s),\tag{2}$$

for all $i_1,\dots,i_k;j_1,\dots,j_s;k=0,\dots,n;s=0,\dots,n-k,$

where the probabilities of meaningless events and coefficients are defined to be zero.

For $k = 0$ and $s = 0$ i_k (and j_s) are not defined, so for example the element $\gamma_{i_k} p\ (0; i_1, \ldots, i_{k-1}; j_1, \ldots, j_s)$ has no meaning, so it is defined to be zero, we have:

$$(\alpha + \sum_{i=1}^{n}(\lambda_i + \gamma_i))p(0;0;0) = \beta p(1;0;0) + \sum_{i=1}^{n} \mu_i p(0;0;i) + \sum_{i=1}^{n} \tau_i p(0;i;0).$$

The system of equations will be simpler if we substitute Equation (2) to Equation (1). Namely, we have

$$\begin{aligned}
(\tau_{i_1} &+ \sum_{\substack{r\neq i_1,\ldots,i_k \\ r\neq j_1,\ldots,j_s}} (\lambda_r + \gamma_r))p(0;i_1,\ldots,i_k;j_1,\ldots,j_s) + \\
&+ \sum_{r=1}^{s} K_{j_r}(0;i_1,\ldots,i_k;j_1,\ldots,j_s) = \\
&= \sum_{\substack{r\neq i_1,\ldots,i_k \\ r\neq j_1,\ldots,j_s}} (\tau_r p(0;r,i_1,\ldots,i_k;j_1,\ldots,j_s) + K_r(0;i_1,\ldots,i_k;j_1,\ldots,r,\ldots,j_s)) + \\
&+ \gamma_{i_k} p(0;i_1,\ldots,i_{k-1};j_1,\ldots,j_s) \\
&+ \sum_{r=1}^{s} \lambda_{j_r} p(0;i_1,\ldots,i_k;j_1,\ldots,j_{r-1},j_{r+1},\ldots,j_s)),
\end{aligned}$$

$$\text{for all } i_1,\ldots,i_k;j_1,\ldots,j_s; k = 0,\ldots,n; s = 0,\ldots,n-k, \tag{3}$$

$$\beta p(1;i_1,\ldots,i_k;j_1,\ldots,j_s) = \alpha p(0;i_1,\ldots,i_k;j_1,\ldots,j_s), \tag{4}$$
$$\text{for all } i_1,\ldots,i_k;j_1,\ldots,j_s; k = 0,\ldots,n; s = 0,\ldots,n-k.$$

The purpose of this paper is to solve this system subject to the normalization condition

$$\sum_{q=0}^{1}\sum_{k=0}^{n}\sum_{s=0}^{n-k} p(q,k,s) = 1,$$

where

$$p(q,k,s) = \sum_{(i_1,\ldots,i_k)\in V_n^k}\ \sum_{(j_1,\ldots,j_s)\in C_{n-k}^s} p(q;i_1,\ldots,i_k;j_1,\ldots,j_s),$$

V_n^k : The set of all $(i_1,\ldots,i_k)$ (as defined above),
C_{n-k}^s : the set of all $(j_1,\ldots,j_s)$ (as defined above).

Such a system of linear equations could easily be solved by standard computational methods, e.g., by Gauss- elimination. But we must not forget that the unknowns are probabilities and therefore - since the state space is very large - the round off errors may have considerable effect on them (see [7,8,9]) and when using computer programs to solve the system of equations, the whole matrix of the equations cannot be stored in a personal computer if $n \geq 3$. It is more efficient to use a recursive computational method to determine the steady-state probabilities, as described in the following section (first it was proposed by Tomkó [10]).

3 The Recursive Solution

Let $\underline{Y}(m)$ be the vector of the stationary probabilities for the states where the operating system is working, there are k failed terminals, and $s = m - k$ jobs are waiting at the CPU $((k = 0, \ldots, m), m = 0, \ldots, n)$. That is,

$$\underline{Y}(m) = \begin{pmatrix} p(0;1,\ldots,m-1,m;0) \\ p(0;1,\ldots,m-1,m+1;0) \\ \vdots \\ p(0;n,\ldots,n-m+1;0) \\ p(0;1,\ldots,m-1;m) \\ p(0;1,\ldots,m-1;m+1) \\ \vdots \\ p(0;n,\ldots,n-m+2;n-m+1) \\ \vdots \\ \vdots \\ p(0;0;n-m+1,\ldots,n) \end{pmatrix}$$

In words, the elements of $\underline{Y}(m)$ are written in lexicographically increasing order of indices

1./ for $k = m$ and $s = 0$,
2./ for $k = m - 1$ and $s = 1$,
⋮
$m + 1$./ for $k = 0$ and $s = m$.

Similarly, let $\underline{Z}(m)$ be the vector of stationary probabilities alike $\underline{Y}(m)$, but for the states, where the CPU is failed. From the definition it is obivous that the dimension of $\underline{Y}(m)$ and $\underline{Z}(m)$ is $\sum_{s=0}^{m} \frac{n!}{(n-m)!s!}$.

Using these notations Equations (3), (4) can be written in matrix form as

$$\begin{aligned} &\underline{Y}(0) = C(0)\underline{Y}(1), && (i) \\ &\underline{Y}(j) = C(j)\underline{Y}(j+1) + D(j)\underline{Y}(j-1),\ j = 1,\ldots,n-1, && (ii) \\ &\underline{Y}(n) = D(n)\underline{Y}(n-1), && (iii) \\ &\underline{Z}(j) = F(j)\underline{Y}(j),\ j = 0,\ldots,n. && (iv) \end{aligned}$$

The dimension of the matrices are / $d(j) = \sum_{s=0}^{j} \frac{n!}{(n-j)!s!}$ / :
$F(j) : d(j) \times d(j)$, $C(j) : d(j) \times d(j+1)$, $D(j) : d(j) \times d(j-1)$.

The elements of all the matrices can be obtained from the Equations (3), (4). For example we can use Equation (4) to obtain the elements of matrix $F(k+s)(k+s = 0,\ldots,n)$: The element $p(1;i_1,\ldots i_k;j_1,\ldots,j_s)$ of $Z(k+s)$ can be obtained from the element $p(0;i_1,\ldots,i_k;j_1,\ldots,j_s)$ of $Y(k+s)$ by multiplying it with $\frac{\alpha}{\beta}$. That is, the matrix $F(k+s)$ contains non-zero elements only in its main diagonal, and this non zero element is the constant value $\frac{\alpha}{\beta}$.

Similarly, we can use the second line of Equation (3) to build the matrix $C(k+s)$, and the third line to determine the matrix $D(k+s)(k+s = 0,\ldots,n)$.

Applying these notations we can state our main result:

Theorem 3.1 *The solution of the Equations (i)-(iv) can be given in the form*

$$\underline{Y}(j) = L(j)\underline{Y}(j-1), j = 1,\dots,n,$$

$$\underline{Z}(j) = F(j)\underline{Y}(j), j = 0,\dots,n, \tag{5}$$

where $L(n) = D(n)$, $L(j) = (I - C(j)L(j+1))^{-1}D(j)$, $j = 1,\dots,n-1$, so the system of equations can be solved uniquely up to a multiplicative constant, which can be obtained from the normalization condition.

Proof. As a starting point of our proof we can observe that equation (iv) is identical with the second equation of (5).
In virtue of equation (iii)

$$\underline{Y}(n) = L(n)Y(n-1).$$

Assuming that $\underline{Y}(j+1) = L(j+1)\underline{Y}(j)$, from (ii) we have

$$\underline{Y}(j) = C(j)L(j+1)\underline{Y}(j) + D(j)\underline{Y}(j-1).$$

By simple calculation we obtain that

$$(I - C(j)L(j+1))\underline{Y}(j) = D(j)\underline{Y}(j-1),$$

$$\underline{Y}(j) = (I - C(j)L(j+1))^{-1}D(j)\underline{Y}(j-1),$$

$$\underline{Y}(j) = L(j)\underline{Y}(j-1),$$

which completes the proof.

Now we can start the recursion with any initial value denoted by $\underline{Y'}(0)$ and the non-normalized $p'(q; i_1,\dots,i_k; j_1,\dots,j_s)$ elements of $\underline{Y'}(m)$, $\underline{Z'}(m)(m = 0,\dots,n)$, can be obtained. We can calculate the steady-state probabilities from $\underline{Y'}(m)$, $\underline{Z'}(m)$ $(m = 0,\dots,n)$, by using the normalization condition as follows

$$\underline{Y}(m) = \frac{\underline{Y'}(0)}{\sum_{q=0}^{1}\sum_{k=0}^{n}\sum_{s=0}^{n-k}\sum_{i_1,\dots,i_k\in V_n^k}\sum_{j_1,\dots,j_s\in C_{n-k}^s} p'(q; i_1,\dots,i_k; j_1,\dots,j_s)}\underline{Y'}(m),$$

$$\underline{Z}(m) = \frac{\underline{Y'}(0)}{\sum_{q=0}^{1}\sum_{k=0}^{n}\sum_{s=0}^{n-k}\sum_{i_1,\dots,i_k\in V_n^k}\sum_{j_1,\dots,j_s\in C_{n-k}^s} p'(q; i_1,\dots,i_k; j_1,\dots,j_s)}\underline{Z'}(m),$$

$m = 0,\dots,n.$

4 Performance Measures

We derive the steady-state characteristics from the steady-state probabilities because the model is too complicated to derive the characteristics directly from the parameters $(n, \alpha, \beta, \ldots)$. Some of these characteristics will be calculated in Table 1-5 (for $n = 4$ and $n = 3$) as examples. We can use these numerical results to investigate how parameters influence the characteristics.

We employ the following usual notation: $\delta(i,j) = 1$, if $i = j$, (and 0 otherwise). The steady-state characteristics:

(i) Mean number of jobs residing at the CPU

$$\overline{n}_j = \sum_{i=0}^{1} \sum_{k=0}^{n} \sum_{s=0}^{n-k} sp(i,k,s).$$

(ii) Mean number of functional terminals

$$\overline{n}_g = n - \sum_{i=0}^{1} \sum_{k=0}^{n} \sum_{s=0}^{n-k} kp(i,k,s).$$

(iii) Mean number of busy terminals

$$\overline{n}_b = \sum_{k=0}^{n} \sum_{s=0}^{n-k} (n-k-s)p(0,k,s).$$

(iv) Utilization of 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 CPU

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

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

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

(vii) Expected response time of jobs for terminal $i/i = 1, \ldots, n/$

$$T_i = \frac{\sum_{q=0}^{1} \sum_{s=0}^{n-k} \sum_{r=1}^{s} \delta(i,j_r)p(q; i_1, \ldots, i_k; j_1, \ldots j_s)}{\lambda_i U_i}.$$

5 Numerical Results

The algorithm described above was implemented on an IBM PC/AT in FORTRAN. We show several examples to illustrate how breakdowns influence the characteristics. The running times were at about 50 seconds for $n = 4$ (Table 1-3), and 2 seconds for $n = 3$ (Table 4-5). If we compare these results to the ones described in [1,10] we can see how scheduling strategy influences the characteristics.

Case 1. Failure free system (See [10]).

$$n = 4 \qquad \alpha = 0.0001 \qquad \beta = 9999.0$$
$$\overline{n}_j = 2.195 \qquad \overline{n}_g = 4.0 \qquad U_{CPU} = 0.906$$

i	λ_i	μ_i	γ_i	τ_i	ω_i	U_i	T_i
1	0.500	0.900	0.0001	9999.0	1.0	0.429	2.658
2	0.400	0.700	0.0001	9999.0	1.0	0.423	3.405
3	0.300	0.600	0.0001	9999.0	1.0	0.452	4.045
4	0.200	0.500	0.0001	9999.0	1.0	0.500	4.998

Table 1.

This case will be the starting point of our investigation. It can be used to test the results and to approximate a failure-free system described in [10]. The difference between these results and the ones in [10] is less than 0.01 for all calculated characteristics. The difference can be derived from the different calculating circumstances (e.g. different computer and programming language). On the other hand this case only approximates a failure-free system.

Case 2. Terminal failure.

$$n = 4 \qquad \alpha = 0.0001 \qquad \beta = 9999.0$$
$$\overline{n}_j = 1.253 \qquad \overline{n}_g = 2.58 \qquad U_{CPU} = 0.666$$

i	λ_i	μ_i	γ_i	τ_i	ω_i	U_i	T_i
1	0.500	0.900	0.3200	0.4500	1.0	0.283	2.133
2	0.400	0.700	0.1700	0.3400	1.0	0.335	2.602
3	0.300	0.600	0.2200	0.5000	1.0	0.336	3.138
4	0.200	0.500	0.1600	0.3000	1.0	0.373	3.842

Table 2.

In this example we can see how terminal failures influence the performance measures. The response times and the number of good terminals are less than in Case 1. That is, the system works as if there were less terminals.

Case 3. CPU failure.

$$n = 4 \qquad \alpha = 0.25 \qquad \beta = 0.45$$
$$\bar{n}_j = 2.195 \qquad \bar{n}_g = 4.0 \qquad U_{CPU} = 0.583$$

i	λ_i	μ_i	γ_i	τ_i	ω_i	U_i	T_i
1	0.500	0.900	0.0001	9999.0	1.0	0.276	4.135
2	0.400	0.700	0.0001	9999.0	1.0	0.272	5.296
3	0.300	0.600	0.0001	9999.0	1.0	0.290	6.292
4	0.200	0.500	0.0001	9999.0	1.0	0.321	7.775

Table 3.

If we compare these results with Case 1, it can be seen, that the failure of the CPU increases the response times and decreases the utilizations, as one can expect.

It seems, that the FIFO (see in [1]) and the PS discipline gives nearly the same results investigating the influence of terminal breakdowns. We got greater CPU utilization in this model than in [1] (for each case). This is the reason why this model is more sensible to the CPU breakdowns (see the response times in Case 3).

Case 4. PPS system (see [3]).

$$n = 3 \qquad \alpha = 0.0001 \qquad \beta = 9999.0$$
$$\bar{n}_j = 1.028 \qquad \bar{n}_g = 3.0 \qquad U_{CPU} = 0.675$$

i	λ_i	μ_i	γ_i	τ_i	ω_i	U_i	T_i
1	0.200	0.400	0.0001	9999.0	1.0	0.508	4.831
2	0.200	0.600	0.0001	9999.0	5.0	0.666	2.498
3	0.200	0.800	0.0001	9999.0	125.0	0.796	1.277

Table 4.

This case can be used to test the algorithm (and the computer program) discussed above. A failure-free PPS system (described in [3]) is approximated by this example. The results are exactly the same as in [3].

Case 5. PPS system with CPU failure.

$$n = 3 \qquad \alpha = 0.1000 \qquad \beta = 0.7000$$
$$\bar{n}_j = 1.028 \qquad \bar{n}_g = 3.0 \qquad U_{CPU} = 0.591$$

i	λ_i	μ_i	γ_i	τ_i	ω_i	U_i	T_i
1	0.200	0.400	0.0001	9999.0	1.0	0.445	5.521
2	0.200	0.600	0.0001	9999.0	5.0	0.583	2.855
3	0.200	0.800	0.0001	9999.0	125.0	0.696	1.459

Table 5.

This case shows, that the more a terminals uses the CPU (or the CPU's queue) the more the response time increases with the CPU failure.

Case 6. PPS system for $n = 5$.

$$n = 5 \quad \alpha = 0.1000 \quad \beta = 0.7000$$
$$\overline{n}_j = 2.278 \quad \overline{n}_g = 4.3 \quad U_{CPU} = 0.777$$

i	λ_i	μ_i	γ_i	τ_i	ω_i	U_i	T_i
1	0.200	0.400	0.1000	0.7000	1.0	0.217	15.6472
2	0.300	0.700	0.0700	0.6000	5.0	0.319	5.836
3	0.250	0.650	0.1500	0.4500	50.0	0.418	2.951
4	0.220	0.600	0.1000	0.7600	15.0	0.406	4.706
5	0.400	0.800	0.1200	0.4400	125.0	0.442	1.747

Table 6.

The program was run for $n = 5$ in this case. This was the largest n value that could be used. The running time was at about 4 minutes.

References

[1] B. Almási and J. Sztrik, A Queueing Model for a Non-Homogeneous Terminal System Subject to Breakdowns, *Computers and Mathematics with Applications* (to appear).

[2] E. Gelenbe and I. Mitrani, *Analysis and Synthesis of Computer Systems*, Academic Press, London, (1980).

[3] J. Sztrik, A probability model for priority processor-shared multiprogrammed computer systems, *Acta Cybernetica 7*, 329-340 (1986).

[4] J. Sztrik, On the heterogeneous machine interference with limited server's availability, *European Journal of Op. Res. 28*, 321-328 (1987).

[5] J. Sztrik and T. Gál, A recursive solution of a queueing model for a multi-terminal system subject to breakdowns, *Performance Evaluation 11*, 1-7 (1990).

[6] R. Goodman, *Introduction to Stochastic Models*, Benjamin/Cummings, California, (1988).

[7] S.S. Lavenberg, Ed., *Computer Performance Modelling Handbook*, Academic Press, New York, (1983).

[8] P.M. Snyder and W.J. Stewart, Explicit and iterative approaches to solving queueing models, *Oper. Res. 33*, 183-202 (1985).

[9] H.C. Tijms, *Stochastical Modelling and Analysis, A Computational Approach*, Wiley and Sons, New York, (1986).

[10] L. Csige and J. Tomkó, The machine interference for exponentially distributed operating and repair times, *Alk. Mat. Lapok 8*, 107-124 (in Hungarian) (1982).

Received September 29, 1992.