INTEGRATED PERFORMANCE/RELIABILITY ANALYSIS OF COMPUTER SYSTEMS AND COMMUNICATION NETWORKS ON THE BASIS OF BCMP-QUEUEING NETWORKS

W. Tedjakusuma, R. Feix, B. Wolfinger
Department of Computer Science
University of Hamburg
Bodenstedtstraße 16
D-2000 Hamburg 50
Federal Republic of Germany

Abstract
In spite of significant progresses in building more reliable and more efficient hardware components for computer and communication systems, e.g. as a consequence of increasing overall system complexity, there is still considerable need for performance and reliability analyses. Starting from a classification of typical errors occurring in data processing and data communication, this paper introduces a error model and demonstrates its use for combined performance/reliability modeling. The approach is based on the application of queueing networks of BCMP-type. Solution techniques for the mathematical analysis of the models are derived and illustrated by way of example.

Keywords: performability analysis, BCMP-queueing networks, error-classification, performance evaluation of computer and communication systems

1. Introduction
Performance and reliability evaluation has become an important aid to support the design and realization of computer systems and communication networks (shortly called system in the sequel). Although the performance and reliability of a system's hardware components have been improved substantially during the past decade, in general, modern systems possess highly complex and sophisticated structures with multiple interactions between their components. Therefore complex models are required for a comprehensive analysis of the total system. An essential property of many contemporary computer systems or networks is fault tolerance, which from the analyst's point of view makes it desirable to study the (reduced) performance of a system when affected by temporary or remanent failures of its components. There are many dependencies between reliability and performance measures. Failures reduce the system's throughput, prolong the sojourn times of tasks (even of those not being affected directly) and cause the system to perform countermeasures. Additionally, the workload of a system has a great impact on the type and frequency of failures [WoS90], so that system designers should base their design decisions on an integrated performance/reliability analysis (cf. performability as introduced by [Mey83]) and thus take advantage of the fact, that workload dependent reliability measures can be evaluated.

As system failures usually occur very rarely, simulative evaluation of realistic system models becomes impractical and analytical evaluation methods should be preferred. However, in order to capture the system behaviour, analytical models have to be quite detailed and flexible and therefore should take into consideration different types of tasks as well as different types of errors and failures.

Product form networks, i.e. BCMP-networks [BCM75], have turned out to be an appropriate tool for system performance modeling [Lav83]. Within this network class different types of tasks may be considered requiring service times and routing chains depending on the task types. Consequently, in this paper we suggest an integrated analysis, which is based on an exact BCMP-network analysis and additionally comprises refined fault phenomena. We restrict our attention here to the case of transient (intermittent) errors, only.

The content of the paper will be as follows: After a detailed classification of errors the paper proposes, for a given queuing network model $Q$ of a system, a stochastic model $\Phi$ of fault phenomena and a modification $Q'$ of $Q$ including $\Phi$. Modifications concern distributions of service times at system components as well as the queueing network's topology. If $Q$ is of the BCMP-type, so is $Q'$. Furthermore, an exact analytical method mainly based on Mean Value Analysis and network decomposition is given; which allows to compute parameters of network $Q'$ so that standard evaluation methods apply [Lav83]. Our analytical approach is illustrated by means of a nontrivial example.
2. error classification and error model

Our view of an (abstract) system is that of a set of components which offer different types of service to competing tasks. The components model hardware and software resources of the system and tasks stand for workload units such as user programs or operating system tasks to be executed or records, packets, frames, etc. to be transferred, depending on the system to be analyzed and on the chosen level of abstraction.

As far as the basic terms 'fault', 'error' and 'failure' are concerned we adopt the concepts and terminology of [Lap85]: A fault of a system component is viewed as the deviation of the component's behavior from the specification of its behavior. Upon occurrence a fault of a component creates a (latent) error, i.e. an incorrect component state.

We assume that each error cycles between two (private) states, a latent and an effective state. The latter indicates, that the error is activated and potentially results in (a component) failure, i.e. the deviation of the currently delivered service from the specified service. A component failure can thus be viewed as the incorrect execution of a task or as the illegal effect an error can have on a task presently being served by the erroneous component.

Faults can be classified in various ways, for example by their persistence or source. Since, however, from the performance analyst's point of view the reduced performance of an erroneous system is the main subject of investigation, we find it more convenient to classify errors rather than faults.

The most important types of errors resulting from our classification are the 'spontaneous', 'self-induced', and 'cross-effective' ones [JeV87], [Sch86]. A component error, which is independent of the system workload, is called spontaneous and task-induced, otherwise. Thus task-induced errors are supposed to be activated (i.e. they switch over into the state 'effective') by tasks visiting the component, where the error resides in. We call a task-induced error self-induced, if tasks activating the error are subject to the resulting failures, whereas a cross-effective error affects tasks at a component different from the error's origin.

The component errors of the different types discussed so far may last in state 'effective' only a short time interval compared to the typical service times required at that component (transient errors) or may last a long time in state 'effective' (permanent errors) before returning to state 'latent' (if ever). With respect to system performance it makes a big difference whether an error is of the transient or of the permanent type.

A transient error (when activated) will disrupt only those tasks currently being served, whereas a permanent error will disrupt tasks at the erroneous component in any case. Throughout this paper we will use the term malfunctions for failures due to transient errors.

Let the symbols sp, te, si, ce denote the error-types spontaneous, task-induced, self-induced and cross-effective, respectively, and let us (more formally) call a (compatible) pair \( (e_1, e_2) \) \( (e_1 \in \{sp, te\}) \) \( (e_2 \in \{si, ce\}) \) an error-type. The definitions e.g. directly imply:

\[
(\neg sp \land (si \lor ce))
\]

If we take into account this implication and combine all spontaneous errors into one single type, we obtain the following (reduced) error-types:

\[
F = \{sp, si, ce\}.
\]

If we now also take into account the time an error is effective, we can differentiate a transient (tr) and a permanent (pm) variant of each type of error. The types \( t \in F \{ce\} \) will be called local.

Let us illustrate the error-types introduced in our classification, by giving some typical examples for computer systems (CS) as well as for communication networks (CN):

\[
sp \land tr:
\]

CS: erroneous disc access (e.g. implied by particles of dust).

CN: short-term existence of a noise signal on a transmission line (affecting only one data packet transmitted).

\[
si \land pm:
\]

CS: break down of a VLSI component (e.g. processor, memory).

CN: long-term existence of unfavourable atmospheric conditions in satellite communications.

\[
si \land tr:
\]

CS: arithmetic overflow as a consequence of an incorrect operand value.

CN: data transfer request containing a wrong destination address.

\[
sp \land pm:
\]

CS: erroneous program implying a run-time error.

CN: loss of circulating control token in a Token Ring network (cf. [Tan89]).

\[
ce \land tr:
\]

As components which offer different types of service to competing tasks, the components model hardware and software resources of the system and tasks stand for workload units such as user programs or operating system tasks to be executed or records, packets, frames, etc. to be transferred, depending on the system to be analyzed and on the chosen level of abstraction.

As far as the basic terms 'fault', 'error' and 'failure' are concerned we adopt the concepts and terminology of [Lap85]: A fault of a system component is viewed as the deviation of the component's behavior from the specification of its behavior. Upon occurrence a fault of a component creates a (latent) error, i.e. an incorrect component state.

We assume that each error cycles between two (private) states, a latent and an effective state. The latter indicates, that the error is activated and potentially results in (a component) failure, i.e. the deviation of the currently delivered service from the specified service. A component failure can thus be viewed as the incorrect execution of a task or as the illegal effect an error can have on a task presently being served by the erroneous component.

Faults can be classified in various ways, for example by their persistence or source. Since, however, from the performance analyst's point of view the reduced performance of an erroneous system is the main subject of investigation, we find it more convenient to classify errors rather than faults.

The most important types of errors resulting from our classification are the 'spontaneous', 'self-induced', and 'cross-effective' ones [JeV87], [Sch86]. A component error, which is independent of the system workload, is called spontaneous and task-induced, otherwise. Thus task-induced errors are supposed to be activated (i.e. they switch over into the state 'effective') by tasks visiting the component, where the error resides in. We call a task-induced error self-induced, if tasks activating the error are subject to the resulting failures, whereas a cross-effective error affects tasks at a component different from the error's origin.

The component errors of the different types discussed so far may last in state 'effective' only a short time interval compared to the typical service times required at that component (transient errors) or may last a long time in state 'effective' (permanent errors) before returning to state 'latent' (if ever). With respect to system performance it makes a big difference whether an error is of the transient or of the permanent type.

A transient error (when activated) will disrupt only those tasks currently being served, whereas a permanent error will disrupt tasks at the erroneous component in any case. Throughout this paper we will use the term malfunctions for failures due to transient errors.

Let the symbols sp, te, si, ce denote the error-types spontaneous, task-induced, self-induced and cross-effective, respectively, and let us (more formally) call a (compatible) pair \( (e_1, e_2) \) \( (e_1 \in \{sp, te\}) \) \( (e_2 \in \{si, ce\}) \) an error-type. The definitions e.g. directly imply:

\[
(\neg sp \land (si \lor ce))
\]

If we take into account this implication and combine all spontaneous errors into one single type, we obtain the following (reduced) error-types:

\[
F = \{sp, si, ce\}.
\]

If we now also take into account the time an error is effective, we can differentiate a transient (tr) and a permanent (pm) variant of each type of error. The types \( t \in F \{ce\} \) will be called local.

Let us illustrate the error-types introduced in our classification, by giving some typical examples for computer systems (CS) as well as for communication networks (CN):

\[
sp \land tr:
\]

CS: erroneous disc access (e.g. implied by particles of dust).

CN: short-term existence of a noise signal on a transmission line (affecting only one data packet transmitted).

\[
si \land pm:
\]

CS: break down of a VLSI component (e.g. processor, memory).

CN: long-term existence of unfavourable atmospheric conditions in satellite communications.

\[
si \land tr:
\]

CS: arithmetic overflow as a consequence of an incorrect operand value.

CN: data transfer request containing a wrong destination address.

\[
si \land pm:
\]

CS: erroneous program implying a run-time error.

CN: loss of circulating control token in a Token Ring network (cf. [Tan89]).

\[
ce \land tr:
\]
CS: mutually interfering signals introducing errors in a concurrent data transfer to/from system disk.
CN: station sending temporarily in an Ethernet network (cf. [Tan89]).

ce \times pm:
CS: program infected by a virus formatting the system disk.
CN: station sending permanently in an Ethernet network.

Now, let C be the set of system components and E the (index-) set of those components, where errors are initiated. Let further, for an index i \in E, \( F_i \subseteq F \) be the set of those fault types which occur at component i. For a transient local error of type \( \tau \) (resp. cross-effective error disrupting component j) originated at i \in E we denote by \( \phi_i(\tau) = 1 \) (resp. \( \phi_i(\{j, \text{ce}\}) = 1 \) the mean time between subsequent visits at state 'effective'. For every i \in E and local type \( \tau \in F_i \) we assume a stochastic (counting-) process \( \phi_i(\text{tr}, \tau) \) generating transient errors of type \( \tau \). For cross-effective errors affecting j we more precisely write \( \phi_i(\{j, \text{ce}\}) \). If component i is origin of permanent faults we denote the corresponding processes by \( \phi_i(\text{pm}, \tau) \) (\( \phi_i(\text{pm}, \{\text{ce}\}) \)). Setting \( \phi_i(\text{tr}, \tau) = \phi_i(\text{pm}, \tau) = \phi_i(\{\text{ce}\}) \) the vector \( \Phi = (\phi_i / i \in E) \) then covers the complete information about error processes and is therefore called the system's error model.

As we consider in this paper malfunctions only, we will drop in the sequel the symbol 'tr' in our notation. Let Q be a closed BCMP-network model of a system S to be analyzed with a set \( I \) of service stations (the nodes of Q), mean service times \( \beta_i = \beta_i(m) \), where m denotes the current population found at node i (i \in I), routing (or transition-) matrix R and a fixed number of jobs visiting the nodes of Q. (For the sake of brevity we will not consider routing chains in the sequel.) We assume a one-to-one-correspondence between I and the set C of components of S. In order to maintain mathematical tractability \( \Phi \) is assumed to fulfill the following properties:

\( (A_1) \) \( \phi_i^{(sp)}(i \in I) \) is an independent family of poisson processes.
\( (A_2) \) The error rates \( \phi_i(\cdot, \tau) \) are constant or at most may depend on the current population of node i (i \in I, \tau \in F_i).

We adopt the following notation: All performance measures of Q are marked by the symbol "\( * \)", if they result from malfunctions due to \( \Phi \); all 'unstared' measures of Q are the original ones, i.e. \( \Phi \) is disregarded. Among the reliability measures we consider are the malfunction rate \( \psi_i(\cdot, \tau) \) (\( \tau \in F_i \)), i.e. the mean number of malfunctions per time unit originated at node i, and the malfunction frequency \( \rho_i(\cdot, \tau) \) (with respect to error-type \( \tau \)). The malfunction frequency is defined by

\[ \rho_i(\cdot, \tau) := \frac{\psi_i(\cdot, \tau)}{D_i} \]

for a local type \( \tau \in F_i(\{\text{ce}\}) \), and

\[ \rho_i(\{\text{ce}\}) := \frac{\psi_i(\{\text{ce}\})}{D_i} \]

for a cross-effective error originated at i and disrupting j, where \( D_i \) is the throughput of node i (i \in I). Then

\[ \rho_i = \sum \rho_i(\cdot, \tau) \]

can be viewed as the mean risk a task runs when it requires service at node i, [JeS85].

With \( u_i \) denoting the utilization of the single server node i \( \in I \) and considering \( (A_1), (A_2) \) we obtain:

\[ (1) \quad \psi_i(\cdot, \tau) = u_i \cdot \frac{\Phi_i}{\beta_i} \quad (\tau \in F_i(\{\text{ce}\})), \]

and

\[ (2) \quad \psi_i(\{\text{ce}\}) = u_i \cdot \frac{\Phi_i(\{\text{ce}\})}{D_i}. \]

If \( k \) denotes a routing chain, we can evaluate the malfunction rate for chain k as follows:

\[ (1)' \quad \psi_{i,k}(\cdot, \tau) = u_{i,k} \cdot \frac{\Phi_i(\cdot, \tau)}{\beta_i} \quad (\tau \in F_i(\{\text{ce}\})), \]

and

\[ (2)' \quad \psi_{i,k}(\{\text{ce}\}) = u_{i,k} \cdot \frac{\Phi_i(\{\text{ce}\})}{D_i}. \]

where \( u_{i,k} \) is the utilization of node i with respect to tasks of chain k.

3. System model modifications

Given an error model \( \Phi \) we are now able to propose modifications of the system model Q including possible countermeasures of the system. Let T be a (tagged) task at node i which finds m tasks upon its arrival and requires \( B_i(m) \) time units of service. We consider two variants of modifications:

Variant 1:
If T is subject to a malfunction while being served, a recovery handling is executed (requiring \( S_i(m) \) time units) and the service of T resumed requiring \( B_i(m) \) units of time. T does not leave the node and thus routing probabilities need not be computed.
anew. Let the malfunction occur $S_i(m)$ time units after the service of $T$ has been taken up. With $\sigma_i(m) := E(S_i(m))$, $\sigma'_i(m) := E(S_i(m) + B_i(m))$ the modified mean service time at node $i$ can then be given as follows:

\begin{equation}
\beta_i'(m) = (1-p_i) \cdot \beta_i(m) + p_i \cdot (\sigma_i(m) + \sigma'_i(m))
\end{equation}

(see fig. 1). The random variable $S_i(n)$ can be viewed as the "age" of the service time. Choosing $\sigma_i'(m) = E(B_i(m)) = \beta_i'(m) = \beta_i(m)$ we obtain the special cases "service of the rest claim" and "service repetition", respectively. Here $E(B_i(m))$ denotes the 2. moment of the service time distribution.

We disregard the case of a new malfunction while the recovery handling is in progress. The probability for such an event is computable but neglectibly small.

\begin{equation}
\beta_i'(m) = (1-p_i) \cdot \beta_i(m) + p_i \cdot (\sigma_i(m) + \sigma'_i(m))
\end{equation}

4. Analysis of the modified system model

As there are many algorithms for the evaluation of BCMP-networks (see the remark above), what remains to be done is the computation of the unknown risk factors $p_i$ ($i \in E$) which depend on the total number $n$ of tasks in $Q$. In the following we give a formula for the calculation of $p_i$ which is recursive with regard to $n$. Therefore, we write $\beta_i'(m,n)$ (more precisely than in (3), (4)) for the mean service time at node $i$, if the current population of this node is $m$ given $n$ tasks in $Q$. In a similar way, given $n$ we denote by $p_i(m,n)$ the probability of finding $m$ tasks at node $i$. Let us assume that tasks at node $i$ are affected by malfunctions of type $\tau$. We consider variant 1 and derive $p_i$ for a local and a cross-effective error:

**Case 1:** $\tau \in F_i \cdot \{ce\}$

Set $p_i:=p_i(n)=p_i^{(tr,sp)}$, $q_i:=q_i^{(tr,sp)}$. For $p_i'(m,n)$ holds [Sau81, Lav83]:

\begin{equation}
p_i'(m,n) = \beta_i'(m,n) \cdot p_i'(m-1,n-1).
\end{equation}

We use the abbreviations: $C_i(m) := \sigma_i(m) + \sigma'_i(m)$, $X_i(n) := q_i \cdot \sum_{m=1}^{n} \beta_i'(m) \cdot p_i'(m-1,n-1)$, $Y_i(n) := q_i \cdot (1+\sum_{m=1}^{n} C_i(m) \cdot p_i'(m-1,n-1))$. From (3) and the definition of $p_i$ we obtain:

\begin{equation}
p_i = X_i(n) \cdot (X_i(n) - Y_i(n))^{-1}
\end{equation}

where $p_i'(m-1,n-1)$ is known from a previous recursion step.

**Case 2:** $\tau = ce$

Let errors be originated at node $j$ and set $p_i:=p_i(n)$ = $p_i^{(ce)}$ and $q_i:=q_i^{(ce)}$. Let $(e_1, ..., e_j)$ be a solution of the traffic equation $eR = e$ for network $Q$ and set $T_j(n) := e_j \cdot X_j(n)$, $X_j'(n) := X_j(n)/\varphi_i$, and $Y_j'(n) := Y_j(n)/\varphi_i$. From (2) and the definition of $p_i(n)$ we obtain:

\begin{equation}
p_i(n) = D^+(n) \cdot T_j(n) \cdot X_j'(n) \cdot (1 + D^+(n) \cdot T_j(n) \cdot (X_j'(n) - Y_j'(n)))^{-1},
\end{equation}

where $D^+(n)$ denotes the network throughput for given load $n$.

Using Little's formula and (4) we get:

\begin{equation}
D^+(n) = n \cdot (V \cdot p_i(n) \cdot e_i \cdot v_i)^{-1},
\end{equation}

where $V := \sum V_i(n)$ (sum over all nodes $i \in I$).
\( V_i(n) := e_i \sum_{m=1}^{n} m \beta_i (m) \cdot p_i^* (m-1,n-1) \), and 
\( \gamma_i(n) := e_i \sum_{m=1}^{n} m \beta_i (m) - C_i(m) \cdot p_i^* (m-1,n-1) \).

With \( \gamma_i(n) := e_i \sum_{m=1}^{n} m \beta_i (m) - C_i(m) \cdot p_i^* (m-1,n-1) \)
using (6), (7) we arrive at the following equation for the network throughput \( D^* = D^*(n) \):

\( D^* D - (V_\pi V - X_\pi \gamma) + D^* (V \cdot n \cdot T_j \gamma) - n = 0. \)

If i is affected by more than one node (say d) causing cross-effective errors, for every n a polynomial of degree d+1 arises for \( D^* \).

5. An example

Let us demonstrate the effectiveness of our integrated approach by means of a modification of a (known) example [JeS85;GLT85]. A computer system is considered consisting of a CPU, a system disk (SD) and a data base disk (DD). The system is assumed to process two types of task streams, a batch and a timesharing stream.

Service strategies are as follows: For station CPU we assume PS (Processor Sharing) and FCFS (First Come First Served) for the disk stations SD and DD. Let the mean service times be \( \beta_{CPU} = 0.27s \), \( \beta_{SD} = 0.4s \), \( \beta_{DD} = 0.55s \). The BCMP-conditions [BCM75] imply that service times at SD and DD must be exponentially distributed. Timesharing tasks are supposed to be generated at terminals attached to the system. So we combine all terminals into one (abstract) station 'Term' with strategy IS (Infinite Server) and mean service time (= mean think time) \( \beta_{Term} = 0.3s \). In the following notation we use the subscripts (or superscripts) 1, 2, 3, 4 for the stations CPU, SD, DD, Term, respectively. A BCMP-network model \( Q \) of the computer system (i.e. the (node x class)-space) is depicted in fig. 3. The batch and timesharing streams are mapped onto 2 disjoint routing chains each having 2 task classes. Class-specific transitions of tasks are expressed by means of arcs labelled by their corresponding routing probabilities. Performance measures of \( Q \) (i.e. station-specific mean delay times \( W \), mean populations \( F \), throughputs \( D \), and utilizations \( U \)) are outlined in table 1 for a workload of \( N = 1,5,10 \) tasks for each routing chain.

### Table 1: performance measures of the example system disregarding errors.

<table>
<thead>
<tr>
<th>N</th>
<th>CPU</th>
<th>SD</th>
<th>DD</th>
<th>TERM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.4027</td>
<td>0.5046</td>
<td>0.8528</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1.5667</td>
<td>1.1678</td>
<td>1.4369</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>3.743</td>
<td>1.5</td>
<td>2.25</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>0.9074</td>
<td>0.5469</td>
<td>0.6078</td>
</tr>
<tr>
<td>5</td>
<td>5.5064</td>
<td>2.1279</td>
<td>2.2853</td>
<td>0.0804</td>
</tr>
<tr>
<td>10</td>
<td>13.5</td>
<td>2.8</td>
<td>3.314</td>
<td>0.086</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>2.0047</td>
<td>1.0841</td>
<td>0.7127</td>
</tr>
<tr>
<td>5</td>
<td>3.3727</td>
<td>1.8222</td>
<td>1.3935</td>
<td>0.258</td>
</tr>
<tr>
<td>10</td>
<td>3.6869</td>
<td>1.8844</td>
<td>1.4748</td>
<td>0.287</td>
</tr>
<tr>
<td>U</td>
<td>1</td>
<td>0.5413</td>
<td>0.434</td>
<td>0.392</td>
</tr>
<tr>
<td>5</td>
<td>0.955</td>
<td>0.729</td>
<td>0.766</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.995</td>
<td>0.754</td>
<td>0.811</td>
<td></td>
</tr>
</tbody>
</table>

### Table 2: malfunction rates and frequencies of the example system when affected by \( \Phi \).

<table>
<thead>
<tr>
<th>N</th>
<th>CPU</th>
<th>SD</th>
<th>DD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.643 \times 10^{-6}</td>
<td>5.352</td>
<td>5.369</td>
</tr>
<tr>
<td>5</td>
<td>1.643 \times 10^{-6}</td>
<td>5.352</td>
<td>5.369</td>
</tr>
<tr>
<td>10</td>
<td>1.643 \times 10^{-6}</td>
<td>5.352</td>
<td>5.369</td>
</tr>
</tbody>
</table>

[Figure 3: (node x class)-space of the example system]
Let us now consider an error model $\phi$ comprising spontaneous and self-induced errors of station CPU with rates (per hour) $\varphi_1(\text{sp}) = 2.16$ and $\varphi_1(\text{si}) = 2.484$, respectively, a cross-effective error of station SD affecting DD with rate $\varphi_3(\text{ce}) = 3.528$, and a spontaneous error of station DD with rate $\varphi_3(\text{sp}) = 1.386$.

Now let $Q'$ be the network resulting from $Q$ when affected by $\phi$. Table 2 contains station-specific malfunction rates $\psi_a := \psi_1(\text{sp})$, $\psi_b := \psi_1(\text{si})$, $\psi_c = \psi_2(\text{ce})$, $\psi_d := \psi_3(\text{sp})$, and the malfunction frequency $p := \rho_2(\text{ce})$ for $N$ chosen as in table 1. Furthermore percentage deviations $\Delta W, \Delta F, \Delta D, \Delta u$ of the performance measures of $Q'$ compared to those of $Q$ are given. Because of the load independent service rates malfunction frequencies due to local errors will not depend on $N$. These values are as follows:

$\rho_1(\text{sp}) = 0.000162, \rho_1(\text{si}) = 0.0001863, \rho_3(\text{sp}) = 0.000385$. The following error correction measures were taken:

- CPU (variant 1): $\beta_1 = (1-p_1)\beta_1 + p_1C_1$, where $C_1 = \sigma_1+\sigma_4 = 5.67$.
- DD (variant 2): The error correction method is assumed to be 'retry' (i.e. the recovery network is bypassed and affected tasks recycle to DD).

Table 2 shows that, although the transient errors occur very rarely, failure rates and frequencies increase clearly, if $N$ is getting large. On the other hand we observe that the CPU-station turns out to be the bottleneck already for small $N$ ($N > 4$) implying an unbalanced system workload. Consequently the mean delay time at CPU increases causing a reduction of system throughput compared to the error-less model.

6. Final remarks

Considering a computer or communication system affected by faults of different types, we started from a queueing model of the system and a stochastic model $\phi$ of error phenomena and elaborated a queueing network $Q'$ incorporating $\phi$. The effect of malfunctions on the network performance was modelled by means of service time extensions or additional (recovery-) networks. The evaluation of $Q'$ involves the risk factor $p_1$ which a task runs when entering the erroneous node $i$. We suggested an exact analytical method for the evaluation of $p_1$ and consequently for $Q'$. As far as we know our approach is totally new and we hope it will turn out to be useful in practice.

A detailed discussion of chain-specific failure rates and frequencies can be found in [FTW90]. For a treatment of open and mixed BCMP-networks the reader is encouraged to contact the authors.

The extension of our method to a more general class of queuing networks (i.e. queuing networks without product-form) will be subject to further research.

References


