

# MODELING FAILURE DEPENDENCIES IN RELIABILITY ANALYSIS USING STOCHASTIC PETRI NETS

Ricardo M. Fricks                      Kishor S. Trivedi  
Center for Advanced Computing and Communication  
Department of Electrical and Computer Engineering  
Duke University  
Durham, NC 27708-0291

## Keywords:

Reliability, Stochastic Petri-nets, Markov Chains.

## Abstract

*In this paper, we study the effect of failure dependencies in reliability models developed using stochastic Petri nets and continuous-time Markov chains. We also suggest a classification structure for the major causes of failure dependencies. Most of the identified categories are then illustrated through variations of a real-time computer model. Of particular interest among the examples developed is the sample model that incorporates human errors.*

## 1 INTRODUCTION

In many reliability models, the lifetime or time-to-failure distributions of system components are assumed to be statistically independent. However, the assumption of independence is easily violated in practice. In case of electronic devices, this is sometimes inappropriate and in the case of mechanical components, the independence assumption is almost always incorrect (Misra 1993). See, for instance, the large proportion of common-cause failures in the U.S. power reactor industry cited in (Dhillon and Singh 1981) (20.58% in 1975) and the alarming statistics of accidents caused by human errors commented in (Modarres 1993) (responsible for 70% of aviation accidents). It is more realistic then to assume some form of statistical dependence among failure and repair events in a system. Extreme

points-of-view even state that “analyses of such systems [redundant and/or diverse systems] that only consider independent failures are a waste of time and resources, are likely to grossly underestimate system unavailability, and will provide, indeed have already provided, misleading indications of the benefits of redundancy and diversity in system design.” (Fleming, Mosleh, and Deremer 1986).

Our purpose in this paper is to explore how these failure dependencies can be represented in **stochastic Petri net** (SPN) models (basic concepts and terminology of stochastic Petri net modeling are reviewed in the appendix of this paper). We start the paper by introducing in Section 2 a taxonomy to classify different types of failure dependencies that can arise in reliability modeling. In order to facilitate this we start a reference reliability model and study variations on it. In Section 3 we describe a hypothetical real-time computer, and develop its reliability model using SPNs. Section 4 discusses how several of the failure dependencies classified in Section 2 can be incorporated in stochastic Petri net models. The exposition is complemented by suitable modifications of the real-time computer reference model. Section 5 concludes the paper comparing the resultant effects of the experiments developed around the hypothetical real-time computer model.

## 2 FAILURE DEPENDENCIES

The taxonomy for failure dependencies we follow in this paper is illustrated in Figure 1. The classification we adopt is based on a comparative analysis of

several others proposed in the reliability engineering literature (Barlow and Proschan 1975; Dhillon and Singh 1981; Gangloff 1974; Henley and Kumamoto 1981; Hokstad 1993; Misra 1993; Modarres 1993; Ramakumar 1993; Smith 1976).

## 2.1 Common-Cause Failures

A **common-cause failure\*** (CCF) is any condition or event that affects several components inducing their simultaneous failure or malfunction. Albeit simple, this definition is difficult to apply because of the problem of deciding whether causes of multiple failures were dependent or not, and the ambiguity of the term “simultaneous” in the definition. Thus, in practice, CCF and *multiple failures* are synonymous, collecting all sources of failure dependencies that are not known, or are difficult to explicitly model. Some types of common-cause failures are:

- *Human Errors*: The failure by humans to carry out a specified task (or the performance of a prohibited action), which could result in damage to equipment and property or disruption of scheduled operations is a major source of common-cause failures, and may be introduced in several stages of the system life cycle.
- *System Environment*: Characteristics of the environment in which the system operates, including natural factors (e.g., flood, lightning, earthquake, fire, tornado, etc.), environmental stresses or shocks, man made hazards, and shared infrastructure, can also generate common-cause failures.
- *Intercomponent*: The failure of a component may affect adversely other components as a result of a chain reaction or domino effect, also known as *cascade failure*. Examples of inter-component dependencies are the functional deficiencies introduced by the inadequacy of designed protective action, and the co-dependence of software and hardware. Failure of software usually does not impact the underlying hardware, so the hardware can continue to execute other software. However, failure of the hardware automatically implies that the software running on the hardware will be unavailable.

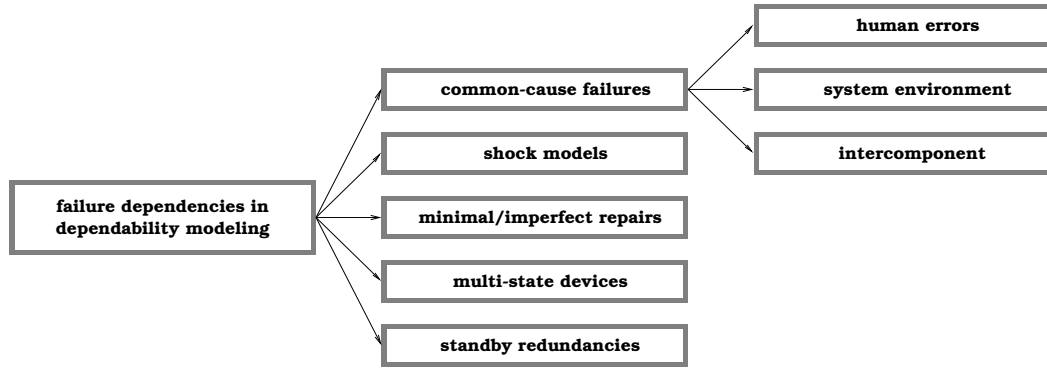
---

\*Some authors prefer the term *common-mode* failure, but the term common-cause seems to be more prevailing in reliability engineering.

## 2.2 Other Failure Dependencies

A dependent failure is the simultaneous failure of more than one component of a system, the probability of which cannot be expressed as the simple product of the unconditional failure probabilities of the individual components. Common-cause failures are archetypal examples of failure dependence. Yet, sometimes failure events are statistically dependent without being related to a common root event:

- *Shock Models* (Barlow and Proschan 1975; Pier-skalla and Voelker 1976; Valdez-Flores and Feldman 1989): A system component is subject to exterior shocks, so that the damage accumulated up to a particular time may result in a failure at that time. The damage accumulates additively until component (or system) replacement or failure. The time between shocks and the damage caused by a shock may depend on the accumulated damage at a given time  $t$ . When the same source of shocks affects several components simultaneously then we have a common-cause failure provoked by the system environment.
- *Minimal or Imperfect Repairs* (Modarres 1993; Valdez-Flores and Feldman 1989): Instead of replacing a failed component by a new one (or to the condition “as-good-as new”), minimal repair restores components that are wearing out only to a certain degree. If the repair of a failed component restores entire system function but the system failure rate remains as it was just before component failure, then the repair is called *minimal*. Similarly, imperfect repairs following failures do not renew system components. These types of repair may result in a trend of increasing failure rates and, therefore, time periods between successive failures are not necessarily independent.
- *Multi-State Devices* (Dhillon and Singh 1981; Ramakumar 1993): Typical system components are binary with respect to dependability characteristics, i.e., either the components are operational or not. However, device such as fluid flow valves and semiconductor diodes have two failure modes instead of a single one, since they may fail in either the open (open-circuit) or closed (short-circuit) mode. Because a three-state device cannot fail simultaneously in both



**Figure 1. Classification of failure dependency modes.**

the open and closed (shorted) modes, the failures are mutually exclusive events. Other types of components may even have multiple exclusive failed states. A relay, for example, may fail by (Henley and Kumamoto 1981): (i) contact problems: stuck open/close; slow in opening/closing; short circuit to ground, to supply, between contacts, and to signal lines; chattering (intermittence); arcing; (ii) coil problems: open/close circuit; low/high resistance; overheating; short circuit: to supply, to contacts, to ground, and to signal lines; and (iii) excessive hysteresis or overmagnetized.

- *Standby Redundancies* (Dhillon and Singh 1981; Ramakumar 1993; Trivedi 1982): Standby sparing involves the dynamic reconfiguration of system components in response to faults. After being detected and located, reconfiguration mechanisms replace the faulty component with a spare or standby. Failure of an operating component thus may cause the selected standby component to be more susceptible to failure and, therefore, component failures are not statistically independent.

### 3 REAL-TIME COMPUTER

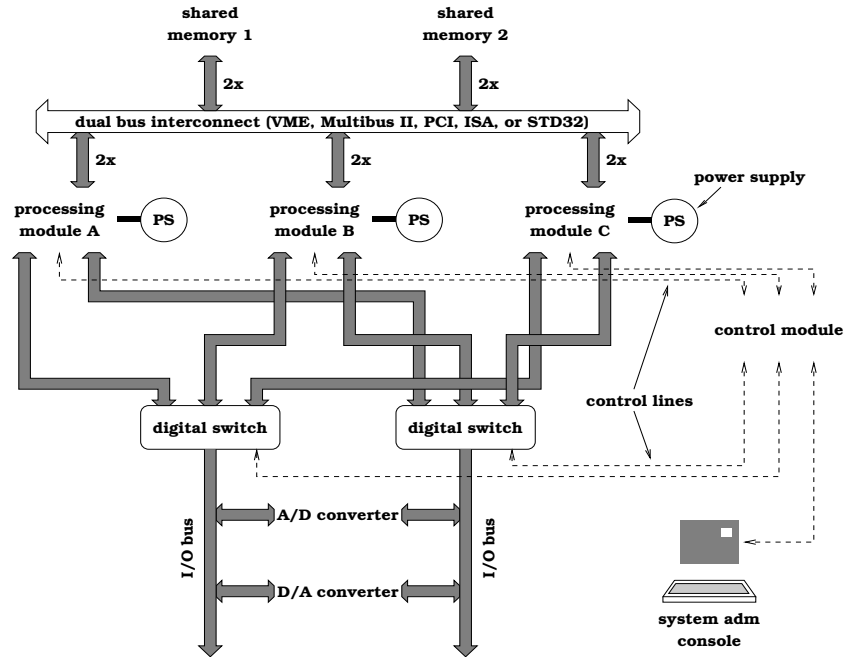
Consider the hypothetical real-time computer shown in Figure 2. The computer has three processing modules, each consisting of processor(s), cache and local memories, power source, interface drives and control circuitry. The local memory of the processing modules stores the real-time operating system kernel, application programs and local data, while two shared memory modules store vital com-

puter information and a real-time database. Processing modules can communicate by exchanging variables stored in shared memory. Data transfer among computer modules and shared memories is possible using a parallel interconnection bus.

An **input/output (I/O)** bus interconnects the processing modules to external interface devices that eventually link the computer to the controlled physical process. **Analog-to-Digital (A/D)** and **Digital-to-Analog (D/A)** converters are connected directly to a dual I/O bus to provide redundant data/control path to the processing modules. Other specialized boards that may be connected to the I/O bus are time standard receiver (GPS, GOES, etc.) and MODEMs to communicate with remote terminal units. Digital switches broadcast all data received from the I/O cards to all processing modules simultaneously.

Additionally, every computer module has two LAN interface cards (not shown in Figure 2) to allow communication with other computers (e.g., database servers or engineering workstations) in a local area network (Ethernet, Token Ring, FDDI, ATM). Conventional peripherals are connected to the computer modules using standard interfaces (e.g., SCSI II) available directly in the boards.

The real-time computer was designed for high-availability applications like data acquisition and control in SCADA systems (Dy-Liacco 1994). High-availability is provided by the implementation of a *pair-and-a-spare* fault-tolerant scheme (Johnson 1989) to operate the processor modules, and parallel redundancy schemes to operate all the other critical



**Figure 2. A hypothetical real-time computer for process control.**

system components (shared memories, I/O bus, and system interconnection bus). The pair-and-a-spare configuration implements a *warm standby sparing* scheme (Henley and Kumamoto 1981) where two processor modules operate online in synchrony, and a spare module remains powered up but off-line until necessary to replace either of the online modules. The advantage of having two online modules instead of a single one, as in traditional standby schemes, is the added error detection capability provided by the continuous comparison of results of both online modules.

Error detection, location and dynamic reconfiguration of the system is coordinated by the control module (shown on the right hand side of Figure 2). The control module is also responsible for selecting which of the online processor modules effectively controls the physical process. The digital switches enforce the directives of the control module: at any time, all processor modules may receive data from the physical process, but only one can send control signals to the process.

The computer system is considered operational as long as there is one operational module of each crit-

ical component: processor module, shared memory, and digital switch. The remaining components (i.e., control module, buses, etc.) are assumed infallible in our models, because their presence would add unnecessary complexity to our exposition.

The hazard and repair rate functions of all modeled component types are assumed to be time independent with rates listed in Table 1. Additional assumptions associated with the model: (i) all system components are fault-free at system start-up; (ii) there is a single infallible repair station for each component module type; (iii) failures and repair activities are statistically independent among different component modules; (v) components of the same type have identical parameters; (vi) repair of online processor modules has priority over the repair of the spare module, i.e., when there is more than one failed processor module, the last to be repaired is the spare processor card; (vii) repair activities are *work conserving*, i.e., they are never idle while there are modules waiting to be repaired; (viii) repaired units are as good as new; and (ix) error detection, location, and system reconfiguration are perfect and instantaneous events. Under these assumptions, we developed the reference SPN models of the three

**Table 1. Component parameters.**

<i>rates</i>		<i>meaning</i>
<i>failure</i>	<i>repair</i>	
$\lambda_o$	$\mu_p$	constant rates of online processor modules
$\lambda_s$	$\mu_p$	constant rates of off-line processor modules
$\lambda_m$	$\mu_m$	constant rates of shared memories
$\lambda_d$	$\mu_d$	constant rates of digital switches

main subsystems of the real-time computer (shown in the left hand side of Figure 3).

The firing rates of all timed transitions are listed in Table 2, associated with the meaning of respective firing events. The sum of component failure rates in the marking dependent transitions<sup>†</sup> **E1**, **E2**, and **E3**, is necessary because the time to the first failure of a group of  $n$  components with independent exponentially distributed lifetimes with parameter  $\lambda_i$  is itself exponentially distributed with parameter  $\sum_{i=1}^n \lambda_i$  (Trivedi 1982). Tokens in places **P1**, **P2**, and **P3**, represent the number of failed component units in each subsystem at a given time. For instance, two tokens in **P1** represent two failed processing modules. The multiple inhibitor arcs in the SPN models enforce the number of component units in each individual subsystem.

When the real-time computer is started up, all of its components are operational. Therefore, the initial marking of the subsystem models has no tokens in any of their places, as shown in Figure 3. The fault-tolerant schemes implemented in the real-time computer define the operational conditions of the subsystems: a pair-and-a-spare scheme (processing modules) is operational as long as there is at least one component module operational, likewise, a parallel scheme (shared memories or digital switches) is operational on condition that at least one module is operational<sup>‡</sup>. Consequently, the real-time computer is operational at any given time  $t$  iff there is at least one processing unit, shared memory, and digital switch. Hence, the instantaneous computer

availability  $A_{rt}(t)$  is given by:

$$\begin{aligned}
 A_{rt}(t) &= \Pr\{\#(\mathbf{P1}) < 3 \wedge \#(\mathbf{P2}) < 2 \wedge \#(\mathbf{P3}) < 2\} \\
 &= \Pr\{\#(\mathbf{P1}) < 3\} \times \Pr\{\#(\mathbf{P2}) < 2\} \\
 &\quad \times \Pr\{\#(\mathbf{P3}) < 2\} \\
 &= A_1(t) \times A_2(t) \times A_3(t),
 \end{aligned} \tag{1}$$

where  $A_1(t)$ ,  $A_2(t)$ , and  $A_3(t)$ , are the separately computed, instantaneous availabilities of the processor modules, shared memories, and digital switches subsystems, respectively.

The solution of an SPN model proceeds by the (automatic) construction of the **Continuous-Time Markov Chain (CTMC)** via the Petri net's reachability graph (Marsan, Conte, and Balbo 1984). For easy reference, the resultant CTMCs are shown on the right hand side of Figure 3, side-by-side to the corresponding SPN models. The states in the state transition diagrams of the subsystems are labelled by the number of tokens in places **P1**, **P2**, and **P3**, respectively. Therefore, the initial states are the leftmost states in the CTMCs (i.e., the states labelled 0), while the failed states are the rightmost.

The infinitesimal generator  $\mathbf{Q}_1$  of the processing modules subsystem,  $\mathbf{Q}_2$  of the shared memories subsystem, and  $\mathbf{Q}_3$  of the digital switches subsystem, are:

$$\mathbf{Q}_1 = \begin{bmatrix} -(2\lambda_o + \lambda_s) & 2\lambda_o + \lambda_s & 0 & 0 \\ \mu_p & -(2\lambda_o + \mu_p) & 2\lambda_o & 0 \\ 0 & \mu_p & -(\lambda_o + \mu_p) & \lambda_o \\ 0 & 0 & \mu_p & -\mu_p \end{bmatrix}$$

$$\mathbf{Q}_2 = \begin{bmatrix} -2\lambda_m & 2\lambda_m & 0 \\ \mu_m & -(\lambda_m + \mu_m) & \lambda_m \\ 0 & \mu_m & -\mu_m \end{bmatrix},$$

<sup>†</sup>A convention followed in our exposition is to refer to the number of tokens in a given place **P** by  $\#(\mathbf{P})$ .

<sup>‡</sup>Note, the similarity of definition and state transition of both schemes, a consequence of our assumption of perfect and instantaneous reconfiguration in both cases.

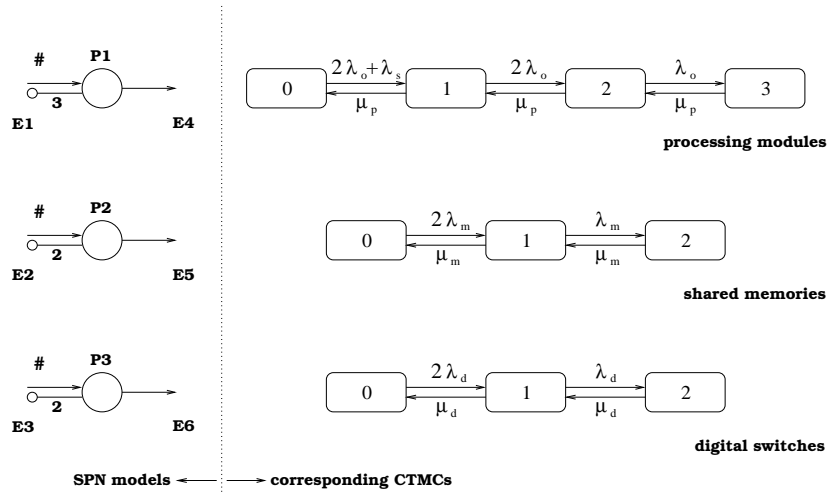


Figure 3. SPN models of three subsystems with corresponding Markov chains.

Table 2. Timed transitions semantics and firing rates.

<i>transition</i>	<i>semantics of firing event</i>	<i>firing rate</i>
<b>E1</b>	failure of a processing module	if $\#(\mathbf{P1}) == 0$ then $2\lambda_o + \lambda_s$ else $[3 - \#(\mathbf{P1})]\lambda_o$
<b>E2</b>	failure of a shared memory	$[2 - \#(\mathbf{P2})]\lambda_m$
<b>E3</b>	failure of a digital switch	$[2 - \#(\mathbf{P3})]\lambda_d$
<b>E4</b>	end-of-repair of a processing module	$\mu_p$
<b>E5</b>	end-of-repair of a shared memory	$\mu_m$
<b>E6</b>	end-of-repair of a digital switch	$\mu_d$

$$\mathbf{Q}_3 = \begin{bmatrix} -2\lambda_d & 2\lambda_d & 0 \\ \mu_d & -(\lambda_d + \mu_d) & \lambda_d \\ 0 & \mu_d & -\mu_d \end{bmatrix}.$$

Finally, we can solve Equation (1) at any given time  $t$  by combining the instantaneous availabilities  $A_1(t)$ ,  $A_2(t)$ , and  $A_3(t)$ , obtained by solving the Kolmogorov differential equation corresponding to each independent CTMC:

$$\frac{d\mathbf{P}_i(t)}{dt} = \mathbf{P}_i(t)\mathbf{Q}_i,$$

with  $i = 1, 2, 3$ , and given the initial state probability vectors

$$\begin{aligned} \mathbf{P}_1(0) &= [ 1 \ 0 \ 0 \ 0 ], \\ \mathbf{P}_2(0) &= [ 1 \ 0 \ 0 ], \quad \text{and} \\ \mathbf{P}_3(0) &= [ 1 \ 0 \ 0 ]. \end{aligned}$$

Naturally, other availability measures (e.g., steady-state and interval) can be also determined given the same elements of the CTMCs (Trivedi 1982).

To complement the presentation of the computer reference model, we introduce an alternative SPN model (shown in Figure 4a) for the pair-and-a-spares configuration of the processing modules. The major differences between the new model and the one shown in Figure 3 are the explicit representation of the standby module and switchover procedure.

Tokens in place **P4** represent online processor modules, while a token in place **P5** means that the standby module is available. The failure of an online processor module is represented by the firing of the timed transition **E7** at the marking dependent rate  $\#(\mathbf{P4}) \times \lambda_o$ . Likewise, the failure event of the spare module is represented by the firing of the timed transition **E8** at the constant rate  $\lambda_s$ . Similarly to the model in Figure 3, tokens in place **P1** represent processor modules waiting to be repaired. As before, the timed transition **E4** firing at constant rate  $\mu_p$  represents the random duration of the repair activity. Once repaired, the modules are restored to either the online (the immediate transition **I2** fires) or offline operational modes (the immediate transition **I3** fires). Transition **I2** has higher priority than transition **I3** to enforce that repaired modules immediately become online if there are less than 2 active processor modules in any circumstance. Finally, the immediate transition **I1** represents the switchover

activity that happens in case one of the online modules fails.

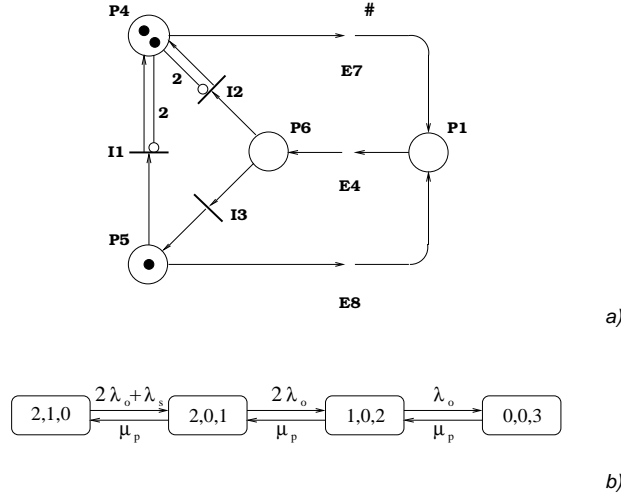
Figure 4b shows the underlying CTMC corresponding to the alternative SPN model. States in this transition diagram are labelled by a triple  $\langle \#(\mathbf{P4}), \#(\mathbf{P5}), \#(\mathbf{P1}) \rangle$ . Observe that although the Petri net models of Figures 3 and 4 are different, their underlying CTMCs are equivalent and, hence, produce the same availability measures. For its compactness, we use the first version of the processing modules subsystem SPN model in the remaining of our discussions. An important aspect to note is that the compact Petri net models of Figure 3 are only possible due to the power of expression of marking dependent timed transitions.

## 4 MODELING USING SPNs

Methodologies for dealing with failure dependencies have been developed since the late 60's, mainly through efforts connected to the nuclear power industry (e.g., at the Safety and Reliability Directorate at UK Atomic Energy Authority, the US Nuclear Regulatory Commission and the Electric Power Research Institute). Results of these efforts include procedural frameworks for CCF analysis using mathematical modeling techniques, such as fault trees and reliability block diagrams. Applying these frameworks certain categories of CCFs, such as environmental impacts and categories of human errors can explicitly be incorporated into stochastic Petri net models, as we present next.

### 4.1 Human Errors

Human reliability analysis puts emphasis on quantification of human errors. Its state-of-art is extensively examined in (Sharit 1993), and a comprehensive portrayal of analysis techniques based on classical reliability engineering appears in (Dhillon 1986). The appeal in adapting classical modeling techniques to the analysis of human reliability lies primarily in the belief that for a number of tasks human performance is sufficiently straightforward in both its observation and measurement to allow accommodation of concepts and techniques usually reserved for hardware components.



**Figure 4. Alternative models for the processing modules subsystem: a) SPN; b) CTMC.**

Employing a classical reliability paradigm for modeling human reliability for time-continuous tasks (such as activities associated with process control applications) requires deriving a time-dependent *human error rate*,  $h_e(t)$  (predicted mainly using THERP - the **Technique for Human Error Rate Prediction** (Sharit 1993)), from which the human reliability function  $R_e(t)$  is obtained:

$$R_e(t) = e^{-\int_0^t h_e(u) du}. \quad (2)$$

Similar to its hardware counterpart, there is also a *human performance correctability function*  $P_c(t)$  for time-continuous tasks. Mathematically,  $P_c(t)$  is the probability that an human error is corrected by time  $t$  (Dhillon 1986):

$$P_c(t) = 1 - e^{-\int_0^t r_c(u) du}, \quad (3)$$

where  $r_c(t)$  is the rate at which human errors can be corrected. Naturally, Equations (2) and (3) hold for any human error rate or correction function. However, if all human error events are statistically independent and occur with constant rates  $h_e(t) = \lambda$  and  $r_c(t) = \mu$ , then a SPN model can be constructed to analyze human reliability.

Human errors can be further classified into *critical* and *noncritical* depending on their effects in the operational state of the system. Noncritical errors induce failure of individual modules, while critical errors have a global impact affecting the whole system at once. A typical example of a noncritical

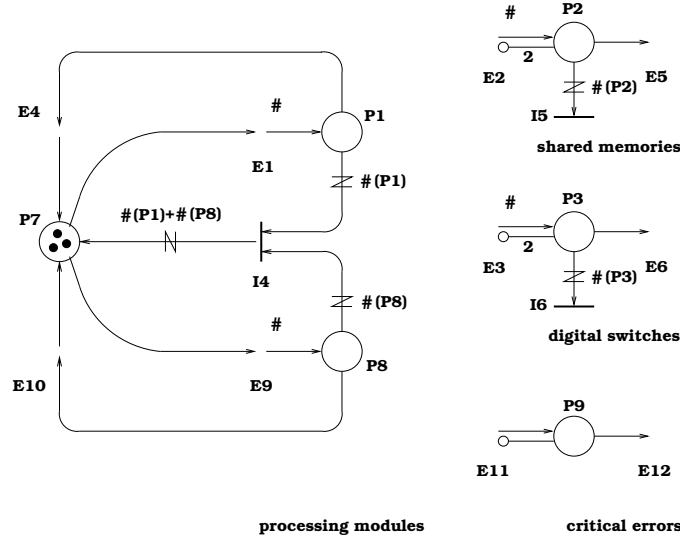
human error is the system administrator sending a wrong configuration command to one of the processing modules. Examples of critical error are fire due to people in a room containing the real-time computer or an erroneous command that produces the shutdown of the whole computer.

Figure 5 illustrates how the real-time computer model can be modified to introduce human errors. Both critical and noncritical types of errors have been introduced. However, it was assumed that noncritical human errors affect only the processing modules. A reasonable assumption considering the strong link that exists between the real-time computer and its system administrator<sup>§</sup>. An interaction that is executed through the system administration console (see Figure 2) and affects specially the real-time operating system running in the processing modules. Shared memories, digital switches, and other components, are also eventually affected by noncritical human errors, but they are not as susceptible as the processors are.

Other assumptions introduced to incorporate human error factors in our model are<sup>¶</sup>: (i) hardware and human events (failures and end-of-repair) are statistically independent; (ii) noncritical errors oc-

<sup>§</sup>We consider only the operational phase of the real-time computer in this example.

<sup>¶</sup>These assumptions are compatible with the assumptions associated with the human reliability models developed in (Dhillon 1986).



**Figure 5. SPN model of the real-time computer incorporating the effect of human errors.**

cur at a constant rate  $\lambda_{nc}$ , while critical errors occur at a constant rate  $\lambda_c$ ; (iii) noncritical human errors are corrected at a constant rate  $\mu_{nc}$  that restores the operational status of the affected processor module; and (iv) critical human errors are only corrected by the complete restoration of the system to its perfect state and happens with a constant rate  $\mu_c$ .

The failure/repair behavior of the subsystems' hardware is preserved as it can be verified comparing Figures 3 and 5. However, new places and transitions (listed in Table 3) were incorporated in the model to properly capture the human influence. The expression to determine the marking dependent rate of transition **E1** also needs to be modified since now processing modules may fail due to human errors or hardware faults. The rate associated with transition **E1** is  $2\lambda_o + \lambda_s$  when there are three tokens in place **P7** and  $\#(\mathbf{P7}) \times \lambda_o$  in all other circumstances.

Other modifications include the addition of places **P7**, **P8**, and **P9**. Tokens in place **P7** represent operational processing modules (either online or off-line), while tokens in **P8** represent failed machines due to human errors, and a token in **P9** indicates that a critical human error has occurred.

The added immediate transitions are necessary to incorporate the critical error event. In the event of a critical human error, the computer will fail, indepen-

dently of the hardware modules states (operational or failed). According to the modeling assumptions, after the critical error is corrected (transition **E12** fires) the system must restart in its initial configuration. This effect can be introduced in a SPN model by adding immediate transitions, variable cardinality arcs, and by associating guards with some timed transitions. Explicitly, places (**P1** and **P8**), **P2**, and **P3**, are connected using variable cardinality input arcs to immediate transitions **I4**, **I5**, and **I6**, respectively. The variable cardinality of the arcs guarantee that all tokens are always removed simultaneously. However, coordination of this flushing activity is also necessary, since it should only happens when a critical human error occurs. For this purpose, guards are associated with the mentioned transitions only enabling them when there is a token in place **P9**. A guard associated with transition **E11** is also necessary to prevent critical human errors from happening when the computer has failed. This guard only enables the timed transition when there is at least one token in place **P7**, and less than two tokens in places **P2** and **P3**.

In the new computer model, the instantaneous availability of the real-time computer is given by

$$A_{rt}(t) = \Pr\{\#(\mathbf{P7}) \geq 1 \wedge \#(\mathbf{P2}) < 2 \wedge \#(\mathbf{P3}) < 2\}. \quad (4)$$

Individual subsystems models are no longer statis-

**Table 3. Firing rates of new transitions incorporated to capture human errors aspects.**

<i>transition</i>	<i>semantics of firing event</i>	<i>firing rate</i>
<b>E9</b>	occurrence of a noncritical error	$\#(P7) \times \lambda_{nc}$
<b>E10</b>	the correction of a noncritical error was completed	$\mu_{nc}$
<b>E11</b>	occurrence of a critical error	$\lambda_c$
<b>E12</b>	the correction of the critical error was completed	$\mu_c$

tically independent, accordingly, Equation (4) cannot be solved following the same procedure of Equation (1). The CTMC corresponding to the global machine model needs to be generated, so that the necessary probabilities can be computed. Among others, an additional measure of interest that can be determined is the expected number of failed processing modules due to non-critical human errors. This measure corresponds to the average number of tokens in place **P8**, i.e.,  $E[\#(\mathbf{P8})]$ .

The transition diagram of the CTMC isomorphic to the SPN model in Figure 5 is considerably more complex than the original one due to three main factors: (i) the introduction of noncritical human errors and corrections in the processing modules model nearly triple the cardinality of its CTMC; and (ii) since subsystems are statistically dependent because of the critical errors, the separate solution of smaller CTMCs is not an option anymore.

For the moment, let us assume the digital switches are no longer part of the computer model<sup>||</sup>. When only noncritical human errors are considered, the transition diagram corresponding to the model we developed is shown in Figure 6. In spite of the inclusion of the human errors, both subsystems are statistically independent. The corresponding Markov chain is then formed by the cross product of the CTMCs corresponding to the processing modules and the shared memories.

To simplify the visualization of the transition diagram in Figure 6, we adopted the following conventions: (i) Each block of states (surrounded by the

<sup>||</sup>We adopt this assumption to avoid unnecessary complexity in our exposition, since everything that we discuss concerning the interaction of the processing modules and shared memories can be easily extended to incorporate imperfect digital switches.

rectangles and labelled by a memory state similar to the ones in the CTMC shown in Figure 3) represents all possible states for the processing modules conditioned to a certain shared memory subsystem state. For instance, the states in the block labelled *memory 1* represent processing module states considering that shared memory module has failed and the other is still operational. (ii) The thick arrows represent common transitions from the states in one block to the corresponding ones in the pointed blocks of states. For instance, besides the regular transitions in its block, state  $\langle 3, 0, 0 \rangle$  in the block *memory 0* has two extra transitions with rates  $2\lambda_m$  and  $\mu_m$ , respectively pointing to and coming from state  $\langle 3, 0, 0 \rangle$  in block *memory 1*. This representation brings clarity and reinforce symmetries when modeling systems composed of various subsystems loosely connected, as it is the case.

States in the transition diagrams corresponding to the processing modules (one inside each block of states) in Figure 6 are labelled by the triple  $\langle \#(\mathbf{P7}), \#(\mathbf{P1}), \#(\mathbf{P8}) \rangle$ . Hence the first digit corresponds to the number of operational processing modules, the second digit is the number of failed modules due to hardware faults, and finally, the third digit gives the number of modules failed due to noncritical human errors. The impact of noncritical human errors can be verified by comparing the transition diagram in one of the blocks with the original CTMC shown in Figure 3.

In Figure 6, states  $\langle 0, 3, 0 \rangle$ ,  $\langle 0, 2, 1 \rangle$ ,  $\langle 0, 1, 2 \rangle$ , and  $\langle 0, 0, 3 \rangle$ , in the blocks labelled *memory 0* and *memory 1*, as well as all the states in block *memory 2*, are the states corresponding to computer failure. On the other hand, the initial state is the state labelled  $\langle 3, 0, 0 \rangle$  in block *memory 0*. If the state of the CTMC at time  $t$  is identified by  $X(t)$  and we use subscripts to identify blocks of

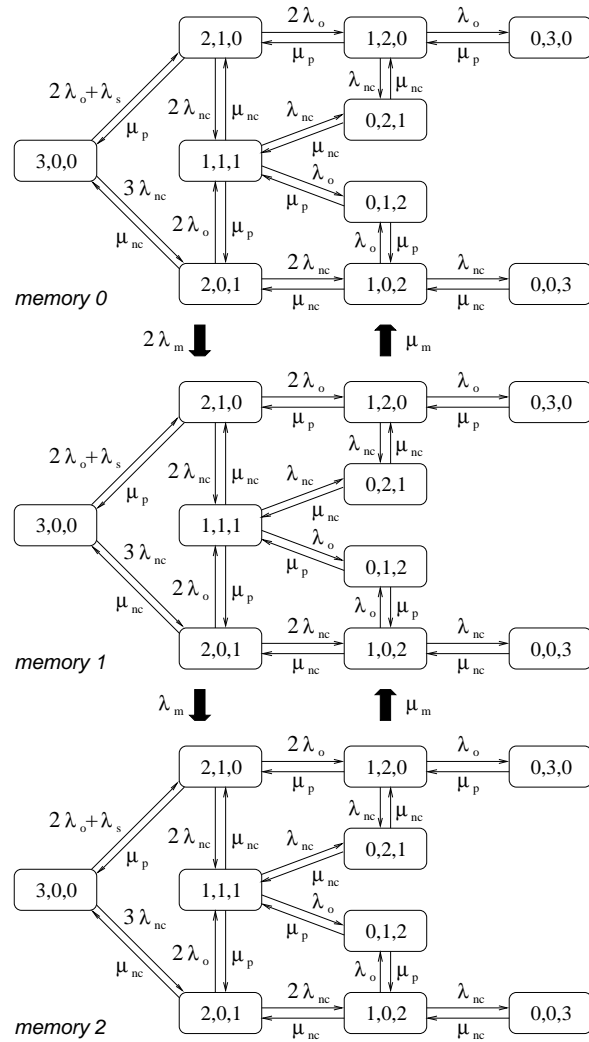


Figure 6. Markov model considering the effect of noncritical human errors.

states, e.g.,  $\langle 3, 0, 0 \rangle_1$  is state  $\langle 3, 0, 0 \rangle$  in block *memory 1*, then we can determine the instantaneous system availability by

$$\begin{aligned}
 A_{rt} = & \sum_{i=0}^1 (\Pr\{X(t) = \langle 3, 0, 0 \rangle_i\} \\
 & + \Pr\{X(t) = \langle 2, 1, 0 \rangle_i\} \\
 & + \Pr\{X(t) = \langle 1, 2, 0 \rangle_i\} \\
 & + \Pr\{X(t) = \langle 2, 0, 1 \rangle_i\} \\
 & + \Pr\{X(t) = \langle 1, 0, 2 \rangle_i\} \\
 & + \Pr\{X(t) = \langle 1, 1, 1 \rangle_i\})
 \end{aligned} \tag{5}$$

Figure 7 sketches the necessary modifications in the transition diagram of Figure 6 to include the effect of human errors. A new state was included in the diagram to represent computer failure caused by a critical human error. All operational states of the system are connected to the critical failure state by transitions with rate  $\lambda_c$  as illustrated. After a critical failure, the computer is restored to the initial state  $\langle 3, 0, 0 \rangle_0$  after an exponentially distributed time with mean  $1/\mu_c$  as also illustrated in Figure 7. The thick gray arrows represent all remaining transitions in the diagram of Figure 6. The transition diagram described corresponds to the CTMC underlying the Petri net model of Figure 3 when the digital switches submodel is removed.

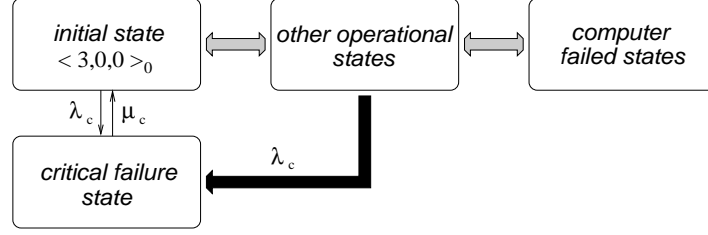
Several other dependability models including human errors can be found in (Dhillon 1986). In reliability analyses involving generally distributed human reliability functions, exact solutions can still be obtained using fault trees or other combinatorial modeling techniques. However, in general, the modeler needs to rely on non-Markovian state space models for availability analyses of repairable systems that include generally distributed  $R_e(t)$  and/or  $P_e(t)$ . This is specially true when repair dependencies also need to be represented in the models (as will be seen later in this section). Examples of non-Markovian models including human errors can be found in (Dhillon and Yang 1993; Dhillon 1986; Dhillon and Rayapati 1988; Yang and Dhillon 1995), where the availability models are solved using the *supplementary variables* technique (Cox 1955). Non-Markovian human reliability models can also be developed using Markov regenerative stochastic Petri nets (Choi, Kulkarni, and Trivedi 1994) or by introducing phase type expansions in regular SPNs

(Cumani 1985).

## 4.2 Other Common-Cause Failures

Other root causes of multiple failures not explicitly modeled can be treated using implicit methods such as parametric models. Failure dependencies modeled using parametric common-cause models include functional deficiencies, inadequate design, manufacturing deficiencies, installation and commissioning errors, maintenance-related errors and aggressive environments. An example of the parametric approach, worth noting because of its wide utilization in structural reliability modeling, is the *beta factor* introduced by Fleming (Fleming 1975). The beta factor  $\beta$  gives the probability that a failure in a specific component causes all components (of a component group or system) to fail. With probability  $1 - \beta$  the failure will involve just the component in question. The beta factor and several other parametric techniques (e.g., *multiple greek letters*, *alpha factor*, and *shock models*) are surveyed in (Fleming, Mosleh, and Deremer 1986; Hokstad 1993; Modarres 1993).

The beta factor is similar in utilization to the *coverage parameter*, although of distinct origin and slightly different semantics. The coverage parameter  $c$  ( $0 \leq c \leq 1$ ), due to Carter et al. (Bouricius, Carter, and Schneider 1969), is a measure of confidence in the error-handling mechanism of a fault-tolerant system. Some authors (Goble 1991; Smith 1991) determine the coverage parameter only based on the fault-detection process, others (Bouricius, Carter, and Schneider 1969; Dugan and Trivedi 1989; Johnson 1989) prefer to consider in its determination all sequential phases of a given fault-handling mechanism (e.g., fault detection, fault location, fault containment, and/or fault recovery). The coverage parameter then may have at least three distinct interpretations. According to Bouricius et al. (Bouricius, Carter, and Schneider 1969), the coverage parameter reflects the conditional probability of successful system recovery given that a fault has occurred. Goble (Goble 1991) defines that the coverage parameter gives the probability that a fault will be detected given that a fault occurred. However, in his reliability calculations, Goble makes the worst-case assumption, i.e., that any fault not covered will cause the entire system failure, which is identical to



**Figure 7. Sketch of the transition diagram including critical human errors.**

the consequence of uncovered failures as described in (Bouricius, Carter, and Schneider 1969).

Despite the nuances in its definition, the impact of coverage in dependability analysis should never be underestimated since dependability measures are very sensitive to small variations in the coverage parameter (see the examples in (Dugan and Trivedi 1989) for instance). The incorporation of coverage parameters in the real-time computer SPN model is illustrated in Figure 8, where also the transition diagrams of the CTMCs associated with each Petri net are also shown. Coverage factors (as well as beta factors) can be easily incorporated in SPN models with the assistance of random switches and variable cardinality arcs.

Most of the places and transitions in the new model preserve the meaning and functions of the model in Figure 3. Tokens in places **P11** and **P13** count the number of active shared memory modules and digital switches, respectively. With the inclusion of these new places, the marking dependent firing rates of timed transitions **E2** and **E3** should be accordingly modified to  $\#(\mathbf{P11}) \times \lambda_m$  and  $\#(\mathbf{P13}) \times \lambda_d$ .

Places **P10**, **P12**, and **P14**, represent decision points in the model and each one is connected to a pair of immediate transitions implementing a random switch (one per subsystem). Transitions **I7**, **I9**, and **I11** (and correspondingly **I8**, **I10**, and **I12**) have distinct firing probabilities (see Table 4) to represent that different subsystems may have distinct coverage parameters. The right hand side of Figure 8 shows the transition diagrams of the CTMCs isomorphic to each subsystem.

Observe that the computer subsystems remain statistically independent among themselves despite

the introduction of the coverage parameters. Therefore, as in Equation (1), the instantaneous availability of the real-time computer at time  $t$  can be computed as

$$\begin{aligned}
 A_{rt}(t) &= \Pr\{\#(\mathbf{P7}) \geq 1 \wedge \#(\mathbf{P11}) \geq 1 \wedge \#(\mathbf{P13}) \geq 1\} \\
 &= \Pr\{\#(\mathbf{P7}) \geq 1\} \times \Pr\{\#(\mathbf{P11}) \geq 1\} \\
 &\quad \times \Pr\{\#(\mathbf{P13}) \geq 1\} \\
 &= A_1(t) \times A_2(t) \times A_3(t).
 \end{aligned} \tag{6}$$

where  $A_1(t)$ ,  $A_2(t)$ , and  $A_3(t)$ , are obtained by solving independent Kolmogorov equations corresponding to Markov chains shown in Figure 8.

### 4.3 Shock Models and Minimal or Imperfect Repairs

Shock models, as well as minimal and imperfect repairs are extensively considered in the works of John McCall (McCall 1965); Pierskalla and Volker (Pierskalla and Voelker 1976); Valdez-Flores and Feldman (Valdez-Flores and Feldman 1989); and Cho and Parlar (Cho and Parlar 1991). These surveys illustrate other types of stochastic processes other than Markov chains are usually necessary to model these types of failure dependencies (e.g., semi-Markov decision processes for minimal repair models and semi-Markov decision, Markov renewal, Lévy, or non-homogeneous Poisson processes for shock models), therefore, we cannot incorporate them by modifying our SPN models.

### 4.4 Multi-State Devices

Three-state devices, the commonest of multi-state devices, arranged in various non-repairable redundant configurations such as series, parallel, bridges, and mixed arrangements, can be modeled following

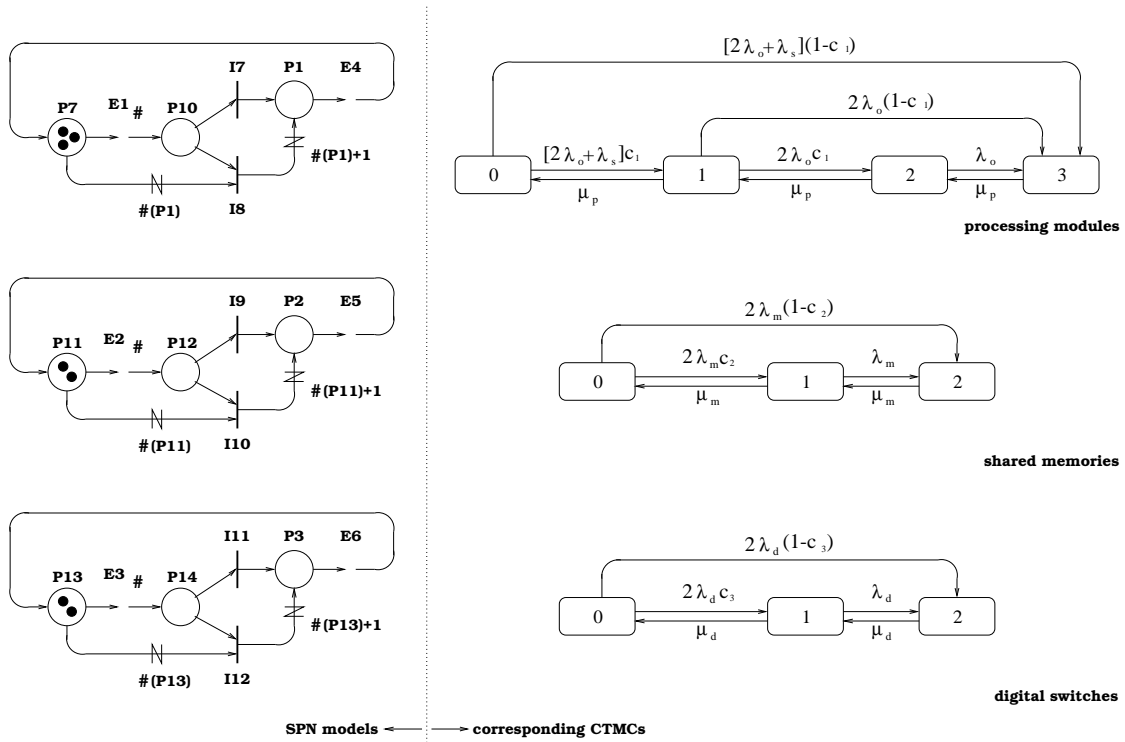


Figure 8. SPN models considering imperfect error-handling mechanism.

Table 4. Characteristics of the added random switches.

<i>subsystem</i>	<i>transition</i>	<i>firing prob.</i>
processing modules	<b>I7</b>	$c_1$
	<b>I8</b>	$1 - c_1$
shared memories	<b>I9</b>	$c_2$
	<b>I10</b>	$1 - c_2$
digital switches	<b>I11</b>	$c_3$
	<b>I12</b>	$1 - c_3$

the combinatorial techniques discussed in (Dhillon and Singh 1981; Dhillon and Rayapati 1986; Gopal, Aggarwal, and Gupta 1978; Jenney and Sherwin 1986; Ramakumar 1993; Singh 1996; Trivedi 1982). State space methods are particularly not suitable for modeling multi-state devices because they need to enumerate all possible system states. For instance, if system components may fail in one of two mutually exclusive states and connections between components are completely reliable, then the entire system state space will have  $3^n$  elements, where  $n$  is the number of components. Clearly, this cardinality limits state space techniques to extremely small systems. Yet, problems involving a large number of components can be solved using Bayes Theorem (Trivedi 1982) associated with recursive techniques like the one described in (Page and Perry 1987), or the hierarchical approach proposed in (Ibe, Howe, and Trivedi 1989).

Suppose the modeled components in the real-time computer can suffer two types of faults: *permanent* and *transient* faults. As a consequence of these fault types, permanent and transient failures may result. Permanent failures remain in existence until a physical action is taken to repair the processor module. On the other hand, transient failures can appear and disappear within a very short period of time. Let us further assume that only online processing modules are affected by transient faults and these faults are corrected by time redundant techniques (e.g., repetition of the operation in error) involving no subsystem reconfiguration or physical repair.

The system availability can still be determined by an SPN model if we assume that all times involved are exponentially distributed. In Figure 9 we show the SPN model of the system considering that in each subsystem: (i) the mean time to the occurrence of a permanent failure and the mean time to complete repairs remain as before; and (ii) the (constant) failure and repair rates of transient failures are  $\sigma_i$  and  $\rho_i$ , respectively, where  $i = o, m, \text{ or } d$ , depending if the parameters are related to the processing modules, shared memories, or digital switches subsystem.

Because of the assumptions made, we need to discriminate the standby unit from the online modules in the SPN model of the processing modules subsystem. Hence, we based our modifications on the

Petri net model in Figure 4, which explicitly identifies the standby component. We also needed to include new timed transitions (listed in Table 5) and extra places **P15**, **P16**, and **P17**. Tokens in these additional places represent module units failed due to transient faults. All added transitions have marking dependent firing rates because we assume that transient events (failures or repairs) are independent among themselves.

An important change made in the SPN of the processing modules subsystem is that the immediate transitions **I1** and **I2** are no longer controlled by multiple inhibitor arcs. Instead, these transitions are guarded so that they are disabled whenever there are two tokens distributed between places **P4** and **P15**. This provision prevents dynamic reconfiguration of the processing modules under transient failure conditions.

The transition diagrams of the corresponding CTMCs on the right hand side of Figure 9 are labelled by duples with the first digit counting the number of failures produced by permanent faults, and the second, the faults caused by transient faults. Thus, in the top diagram, the labels correspond to the number of tokens in places **P1** and **P15**, respectively. In the shared memories subsystem CTMC, labels are  $\langle \#(\mathbf{P2}), \#(\mathbf{P16}) \rangle$ , while in the remaining subsystem, states are labelled  $\langle \#(\mathbf{P3}), \#(\mathbf{P17}) \rangle$ .

Similarly to the coverage modeling example, the instantaneous subsystem availability is determined by solving Equation (6).

## 4.5 Standby Redundancy

Henley and Kumamoto (Henley and Kumamoto 1981) suggest the identification of three distinct phases when analyzing standby redundancy schemes: standby, operation, and repair. Based on component failure characteristics during these phases, standby redundancy can be classified into hot, warm, or cold. The *hot standby* modality consists of statistically independent components since failure rates are identical for the operational and standby phases. On the other hand, the other two modalities induce mutual dependence among failure events associated to components. In the *warm standby*, standby components can fail, but with an

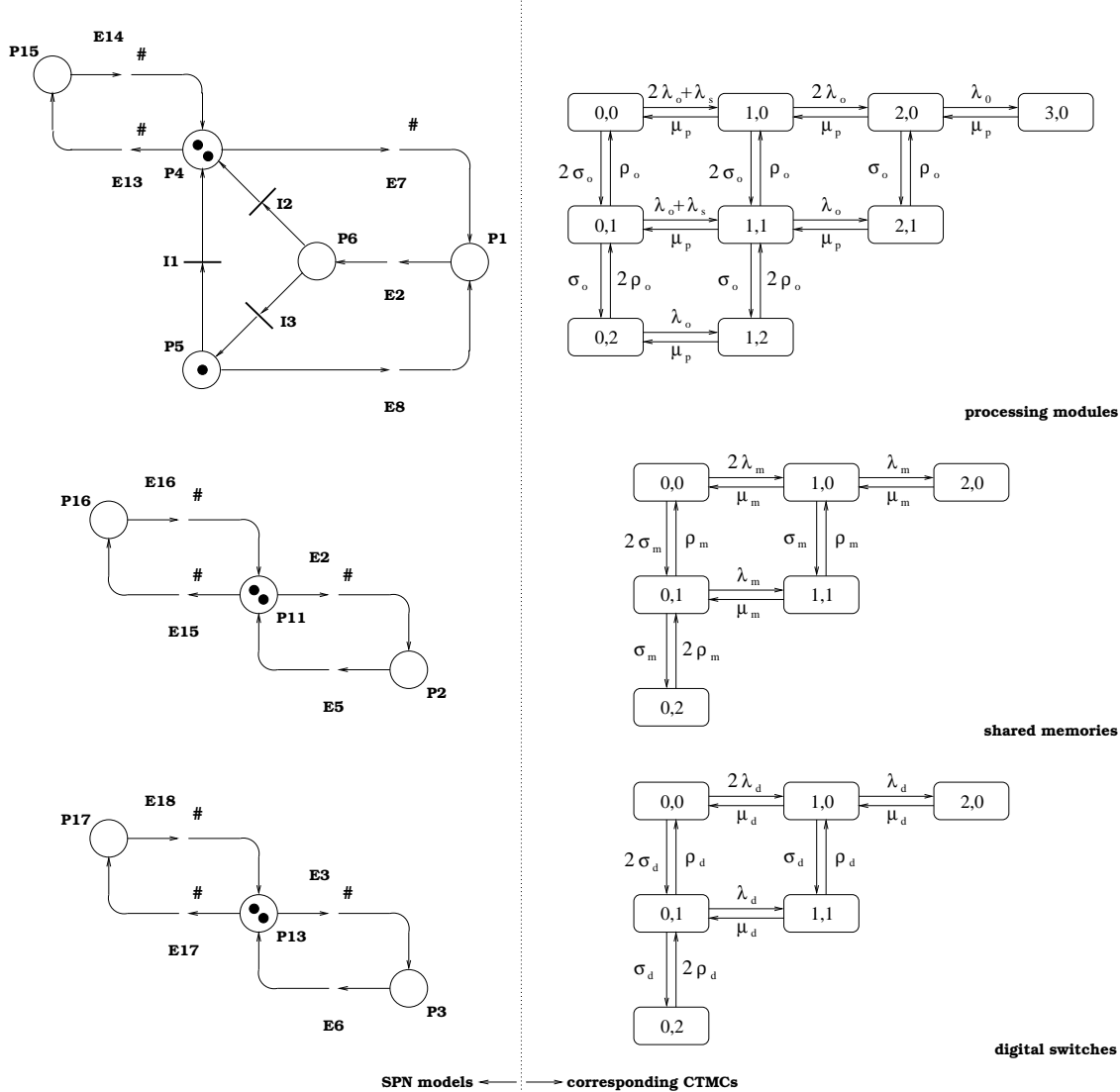


Figure 9. SPN models considering components with two distinct failure modes.

Table 5. Firing rates of new transitions in the model considering transient faults.

<i>transition</i>	<i>semantics of firing event</i>	<i>firing rate</i>
E13	occurrence of a transient fault in a processing module	$\#(P4) \times \sigma_o$
E14	recovery of a transient fault in a processing module	$\#(P15) \times \rho_o$
E15	occurrence transient fault in a shared memory module	$\#(P11) \times \sigma_m$
E16	recovery of a transient fault in a shared memory module	$\#(P16) \times \rho_m$
E17	occurrence of a transient fault in a digital switch	$\#(P12) \times \sigma_d$
E18	recovery of a transient fault in a digital switch module	$\#(P17) \times \rho_d$

smaller failure rate than operational components (online components). In the *cold standby*, standby components do not fail, since they are not powered up. When an online component fails, the failure rate of one (of possibly many) standby components increases, creating the failure dependency.

The pair-and-a-spare configuration of processing modules in the original system model is operating according to a warm standby redundancy scheme. The warm standby or spare module has a failure rate  $\lambda_s$  different from  $\lambda_o$ , the failure rate of the online processor modules, with  $\lambda_s < \lambda_o$ . The failure dependency is trivial in this case since the failure rate of a processing module depends on its role in the subsystem.

To examine the failure/repair behavior of the other standby strategies, we developed two alternative SPN models to the processing modules subsystem. The top SPN (and corresponding CTMC) of Figure 10 represents a hot-standby configuration. When modelling the hot standby scheme we consider that the failure rate of the standby module is identical to the failure rate of online processors, i.e.,  $\lambda_s = \lambda_o$ . Hence, the firing rate associated to **E1** is given by  $[3 - \#(\mathbf{P1})] \times \lambda_o$ , instead of the more complex expression in Table 2.

The CTMC underlying the SPN model of the hot standby scheme can be described by the transition diagram shown in Figure 10. States in this diagram are identified by the number of failed modules, i.e., number of tokens in place **P1**.

On the other hand, the representation of the cold standby strategy needs the alternative SPN model shown in Figure 4. In the cold standby model of Figure 10, a timed transition **E19** replaces the immediate transition **I1** previously representing an instantaneous switchover of the standby module. An assumption certainly not appropriated in the cold standby context, since in this situation the standby module is kept powered off until necessary. For simplicity we assume the procedures necessary to power up and synchronize the standby module with the online modules (if they exist) takes a random amount of time well described by an exponential distribution with parameter  $\alpha^{**}$ . Another reasonable as-

sumption made in the cold standby model is that the standby unit cannot fail while powered-off, i.e.,  $\lambda_s = 0$ .

## 5 CONCLUSIONS

Table 6 shows a comparison of the introduction of failure dependencies in the real-time computer model. Entries in this table refer to the examples previously discussed, the respective size (in terms of number of states and transitions) of the CTMC underlying the stochastic Petri net model, and an observation stating if the three subsystems in the model remained statistically independent (*s-independent*) or not. The numbers in parentheses following the cases where s-independence is preserved refers to the cardinality of the processing modules, shared memories, and digital switches subsystems, in this particular order.

For easy assessment of impacts, we organized the table entries in ascending order of state space cardinality (when jointly solved all subsystems in a single model). Although strongly correlated to the particular example analyzed, the table illustrates how influential the dependencies can be in reliability models and should never be ignored. It can be verified, for instance, that the introduction of human errors (a very important issue) nearly triples the size of the model.

The paper also shows that due to extension of the effects, there is no feasible alternative to anticipate resulting changes in a reference model other than introducing the failure dependencies. For instance, one possibility of estimating the impact of failure dependencies in a reference model would be by applying the specific branch of *perturbation theory* of Markov chains (Schweitzer 1968) and bound the changes in particular measures. An alternative that may provide interesting results when analyzing cases where the dimension of the infinitesimal generator matrix is preserved (e.g., variation in the coverage parameter). However, even in this case caution would be required since one of the best results of perturbation theory oriented to CTMCs (see (Ramesh and Trivedi 1993)) suggests error bounds in proba-

---

\*\*A more realistic deterministic distribution for the switchover of the standby component can be modeled

---

by replacing the timed transition **E19** by a phase type approximation.

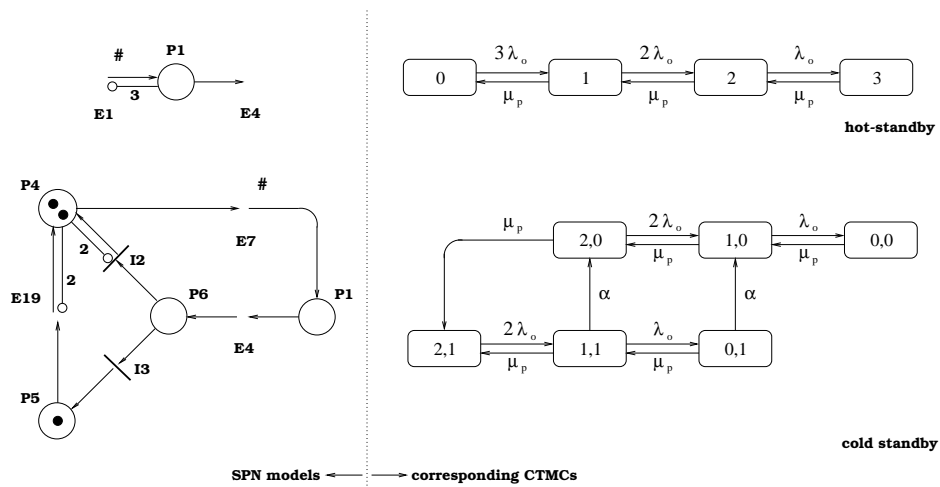


Figure 10. SPN models with hot and cold standby modes.

Table 6. Effects of the failure dependencies in the CTMC model of the real-time computer example.

<i>failure dependency example</i>	<i>state space</i>	<i>transitions</i>	<i>s-independence</i>
warm standby ( <i>reference</i> )	36	150	preserved (4x3x3)
hot standby	36	150	preserved (4x3x3)
coverage parameter	36	192	preserved (4x3x3)
critical human error	37	163	destroyed
cold standby	54	243	preserved (6x3x3)
non-critical human error	90	456	preserved (10x3x3)
non-critical & critical h. errs.	97	501	destroyed
two failure modes	324	2088	preserved (9x6x6)

bility measures after a perturbation in the infinitesimal generator matrix as a function of the model stiffness, a parameter some times problematic when analyzing realistic reliability models.

## References

- Barlow, R. E. and F. Proschan (1975). *Statistical Theory of Reliability and Life Testing - Probability Models*. New York, NY, USA: Holt, Rinehart and Winston.
- Bouricius, W. G., W. C. Carter, and P. R. Schneider (1969). Reliability modeling techniques for self-repairing computer systems. In *Proceedings of the 24th Annual ACM Nat. Conf.*, pp. 295–309.
- Cho, D. I. and M. Parlar (1991, March 6). A survey of maintenance models for multi-unit systems. *European Journal of Operational Research* 51(1), 1–23.
- Choi, H., V. G. Kulkarni, and K. S. Trivedi (1994). Markov regenerative stochastic Petri nets. *Performance Evaluation* 20, 337–357.
- Ciardo, G., A. Blakemore, P. F. Chimento, J. K. Muppala, and K. S. Trivedi (1992). Automatic generation and analysis of markov reward models using stochastic reward nets. In C. Meyer and R. Plemmons (Eds.), *Linear Algebra, Markov Chains, and Queuing Models, IMA Volumes in Mathematics and its Applications*, Volume 48, Heidelberg, Germany. Springer-Verlag.
- Cox, D. R. (1955). The analysis of non-Markovian stochastic processes by the inclusion of supplementary variables. *Proceedings of the Cambridge Philosophical Society* 51(3), 433–441.
- Cumani, A. (1985, July 1–3). ESP - a package for the evaluation of stochastic Petri nets with phase-type distributed transition times. In *Proceedings of International Workshop on Timed Petri Nets*, Torino, Italy, pp. 144–151.
- Dhillon, B. S. (1986). *Human Reliability With Human Factors*. New York, NY, USA: Pergamon Press.
- Dhillon, B. S. and S. N. Rayapati (1986). A method to evaluate reliability of three-state device networks. *Microelectronics and Reliability* 26(3), 535–554.
- Dhillon, B. S. and S. N. Rayapati (1988). Human error modeling of parallel and standby redundant systems. *International Journal of Systems Science* 19(4), 589–611.
- Dhillon, B. S. and C. Singh (1981). *Engineering Reliability: New Techniques and Applications*. New York, NY, USA: John Wiley & Sons.
- Dhillon, B. S. and N. Yang (1993, August). Availability of a man-machine system with critical and non-critical human error. *Microelectronics and Reliability* 33(10), 1511–1521.
- Dugan, J. B. and K. S. Trivedi (1989, June). Coverage modeling for dependability analysis of fault-tolerant systems. *IEEE Transactions on Computers* 38(6), 775–787.
- Dy-Liacco, T. E. (1994, October). Modern control centers and computer networking. *IEEE Computer Applications in Power* 7(4), 17–22.
- Fleming, K. N. (1975). A redundant model for common mode failures in redundant safety systems. In *Proceedings of the Sixth Pittsburgh Annual Modeling and Simulation Conference*, Pittsburgh, PA, USA, pp. 579–581.
- Fleming, K. N., A. Mosleh, and R. K. Deremer (1986). A systematic procedure for the incorporation of common cause events into risk and reliability models. *Nuclear Engineering and Design* 93, 245–273.
- Gangloff, W. C. (1974). Common mode failure analysis. *IEEE Transactions on Power Apparatus and Systems*, 27–30.
- Goble, W. M. (1991). High availability systems for safety and performance - the “coverage” factor. *ISA Transactions* 30(4), 45–49.
- Gopal, K., K. K. Aggarwal, and J. S. Gupta (1978, August). Reliability analysis of multistate device networks. *IEEE Transactions on Reliability R-27*(3), 233–236.
- Henley, E. J. and H. Kumamoto (1981). *Reliability Engineering and Risk Assessment*. Englewood Cliffs, NJ, USA: Prentice-Hall.
- Hokstad, P. (1993). Common cause and dependent failure modeling. In K. B. Misra (Ed.), *New Trends in System Reliability Evaluation*, Amsterdam, The Netherlands, pp. 411–444. Elsevier Science Publishers.
- Ibe, O. C., R. C. Howe, and K. S. Trivedi (1989, April). Approximate availability analysis of

- vaxcluster systems. *IEEE Transactions on Reliability* 38(1), 146–152.
- Jenney, B. W. and D. J. Sherwin (1986, December). Open & short circuit reliability of systems of identical items. *IEEE Transactions on Reliability R-35*(5), 532–538.
- Johnson, B. W. (1989). *Design and Analysis of Fault-Tolerant Digital Systems*. Reading, MA, USA: Addison-Wesley Publishing Company.
- Marsan, M. A., G. Conte, and G. Balbo (1984, May). A class of generalized stochastic Petri net for the performance evaluation of multiprocessor systems. *ACM Transaction on Computer Systems* 2(2), 93–122.
- McCall, J. J. (1965). Maintenance policies for stochastically failing equipment: A survey. *Management Sciences* 11, 493–524.
- Misra, K. B. (1993). *New Trends in System Reliability Evaluation*. Amsterdam, The Netherlands: Elsevier.
- Modarres, M. (1993). *What Every Engineer Should Know About Reliability and Risk Analysis*. New York, NY, USA: Marcel Dekker.
- Molloy, M. K. (1982, September). Performance analysis using stochastic Petri nets. *IEEE Transaction on Computers C-31*(9), 913–917.
- Page, L. B. and J. E. Perry (1987). Reliability of networks of three-state devices. *Microelectronics and Reliability* 27(1), 175–178.
- Peterson, J. L. (1977, September). Petri nets. *Computing Surveys* 9(3), 223–252.
- Pierskalla, W. P. and J. A. Voelker (1976). A survey of maintenance models: The control and surveillance of deteriorating systems. *Naval Research Logistics Quarterly* 23, 353–388.
- Ramakumar, R. (1993). *Engineering Reliability: Fundamentals and Applications*. Englewood Cliffs, NJ, USA: Prentice Hall.
- Ramesh, A. and K. Trivedi (1993, May 10–14). On the sensitivity of transient solution of Markov models. In *Proc. 1993 ACM SIGMETRICS Conf. Measurement and Modeling of Computer Systems*, Santa Clara, CA, USA, pp. 122–134.
- Schweitzer, P. J. (1968). Perturbation theory and finite Markov chains. *Journal of Applied Probability* 5, 401–413.
- Sharit, J. (1993). Human reliability modeling. In K. B. Misra (Ed.), *New Trends in System Reliability Evaluation*, Amsterdam, The Netherlands, pp. 369–410. Elsevier Science Publishers.
- Singh, B. (1996, February). Global reliability of three-state systems. *Microelectronics and Reliability* 36(2), 241–242.
- BOOK: Smith, C. O. (1976). *Introduction to Reliability in Design*. New York, NY, USA: McGraw-Hill.
- JOURNAL: Smith, S. E. (1991). Fault coverage in plant protection systems. *ISA Transactions* 30(4), 51–66.
- Trivedi, K. S. (1982). *Probability & Statistics with Reliability, Queuing, and Computer Science Applications*. Englewood Cliffs, NJ, USA: Prentice-Hall.
- Valdez-Flores, C. and R. M. Feldman (1989). A survey of preventive maintenance models for stochastically deteriorating single-unit systems. *Naval Research Logistics Quarterly* 36, 419–446.
- Yang, N. and B. Dhillon (1995, July). Stochastic analysis of a general standby system with constant human error and arbitrary system repair rates. *Microelectronics and Reliability* 35(7), 1037–1045.

## Appendix: Stochastic Petri Nets

A **Petri net** (PN) (Peterson 1977) is a versatile modeling paradigm that, among other applications, allows the analysis of discrete-event systems with concurrent or parallel events. The pictorial representation of a PN is a bipartite directed graph with places (drawn as circles) and transitions (drawn as bars) that model the static properties of the system. Directed arcs in the graph connect places to transitions (called input arcs) and transitions to places (called output arcs). A multiplicity factor may be associated with these arcs to simplify system description.

Dynamic system properties are modeled by the execution of the PN, which is controlled by the position and movement of markers. These markers or tokens (drawn as solid dots) can only reside inside

places and move around the net by the firing of transitions. Firing rules govern the movement of tokens and enabling conditions for transitions. Enabled transitions may fire, removing one or more tokens from their input places (places linked to input arcs of the transition) and adding tokens to output places (places linked to output arcs of the transition). The firing of a transition is an atomic action possibly resulting in a new marking of the PN, enabling or disabling other transitions. Each distinct marking obtained by the execution of a PN corresponds to an individual state of the PN.

After the original conception, new descriptive elements were proposed to increase the fundamental modeling and/or decision power of ordinary Petri nets. An *inhibitor arc* is always directed from a place to a transition and has a circle rather than an arrowhead at the transition to distinguish from other regular arcs. The firing rule for the transition is changed such that the transition is disabled if there is at least one token present in the corresponding inhibiting input place. A multiplicity factor may be associated with (input, output or inhibitor) arcs to simplify system description. The number of arcs connecting a place to a transition (a transition to a place) is called the *multiplicity* of that arc. When the multiplicity of an arc is more than one, a small number (equal to the multiplicity) is placed next to the arc. An inhibitor arc with multiplicity  $k$  disables the transition when there are at least  $k$  tokens in its connected input place. Another simplifying extension is the assignment of *priorities* to transitions. Enabled transitions fire according to their priorities: the highest priority transition fires first, then the priority immediately below fires, etc.

Less common logical extensions are guards and variable cardinality arcs. *Guards* or *enabling functions* are boolean functions that can be associated with transitions. If in a given marking of the PN, the guard of transition  $t_i$  evaluates to FALSE then transition  $t_i$  is disabled, otherwise the state of the transition is not modified. *Variable cardinality arcs* allow marking dependent definition of the multiplicity of arcs in a PN. These constructs are very valuable to represent events like the flushing of tokens in places, and increase the compactness of description of PNs.

Since its introduction, PNs and its early ex-

tensions have proved to be an invaluable formalism for modeling systems that exhibit asynchronous and concurrent activities. Although very powerful, the initial abstractions provided were not complete enough to capture elements indispensable for the performance and reliability evaluation. Processes modeled by PNs are characterized by lack of the concept of time as a parameter. No assumption was made about the holding times in states, or the relation between these times. New extensions became apparent in order to allow for the specification of the duration of system activities. In (Molloy 1982), Molloy introduced the *stochastic Petri net* paradigm by assigning to each transition an exponentially distributed firing time (for continuous systems) or a geometrically distributed firing time (for discrete systems). The set of reachable markings from a given initial marking became a stochastic process (called *marking process*), which was shown to be equivalent to a **Continuous Time Markov Chain** (CTMC).

**Generalized Stochastic Petri Nets** (GSPNs) (Marsan, Conte, and Balbo 1984) are SPNs that allow two types of transitions: immediate and timed. The *timed transitions* (drawn as hollow rectangles if the firing rate is constant, and as hollow rectangles associated with the symbol “#” if the firing rate is marking dependent) still fire after a random time governed by an exponential distribution as in the SPN case. On the other hand, *immediate transitions* (drawn as thin bars) once enabled fire immediately, like the original PN transitions. Firing rates of timed transitions may depend on the GSPN marking, and immediate transitions have priority to fire over timed transitions. The concept of *random switches* was introduced to solve conflicts among simultaneously enabled immediate transitions in a probabilistic fashion. A probability mass function (called *switching distribution* and possibly marking-dependent) is associated to the competing transitions, according to each only one immediate transition is selected to fire.

The GSPN model extended to allow variable cardinality arcs, guards and reward rate definitions at the net level leads to **Stochastic Reward Nets** (SRNs) (Ciardo, Blakemore, Chimento, Muppala, and Trivedi 1992) that we use in this paper.