Chapter 14

# A Petri Net-Based Approach to the Quantification of Data Center Dependability

Gustavo Callou, Paulo Maciel, Dietmar Tutsch, Julian Araújo, João Ferreira and Rafael Souza

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/47829

## 1. Introduction

Data center availability and reliability have accomplished greater concern due to increased dependence on Internet services (e.g., Cloud computing paradigm, social networks and e-commerce). For companies that heavily depend on the Internet for their operations, service outages can be very expensive, easily running into millions of dollars per hour [15]. A widely used design principle in fault-tolerance is to introduce redundancy to enhance availability. However, since redundancy leads to additional use of resources and energy, it is expected to have a negative impact on sustainability and the associated cost.

Data center designers need to verify several trade-offs and select the feasible solution considering dependability metrics. In this context, formal models (e.g., Stochastic Petri nets and Reliability Block Diagrams) are important to provide estimates before implementing the data center system. Additionally, a growing concern of data center designers is related to the identification of components that may cause system failure as well as systems parts that must be improved before implementing the architecture.

In this work, we propose a set of formal models for quantifying dependability metrics for data center power infrastructures. The adopted approach takes into account a hybrid modeling technique that considers the advantages of both stochastic Petri nets (SPN) [22] and reliability block diagrams (RBD) [10] to evaluate system dependability. An integrated environment, namely, ASTRO [20] has been developed as one of the results of this work to automate dependability evaluation of data center architectures.

## 2. Preliminaries

This section briefly touches some fundamental concepts as a basis for a better understanding of this work.

## 2.1. Petri nets

Petri nets (PN) were introduced in 1962 by the PhD dissertation of Carl Adams Petri [16], at Technical University of Darmstandt, Germany. The original theory was developed as an approach to model and analyze communication systems. Petri Nets (PNs) [14] are a graphic and mathematical modeling tool that can be applied in several types of systems and allow the modeling of parallel, concurrent, asynchronous and non-deterministic systems. Since its seminal work, many representations and extensions have been proposed for allowing more concise descriptions and for representing systems feature not observed on the early models. Thus, the simple Petri net has subsequently been adapted and extended in several directions, in which timed, stochastic, high-level, object-oriented and coloured nets are a few examples of the proposed extensions.

## 2.2. Place-Transition nets

Place/Transition Petri nets are one of the most prominent and best studied class of Petri nets, and it is sometimes called just by Petri net (PN). A marked Place/Transition Petri net is a bipartite directed graph, usually defined as follows:

**Definition 2.1. (Petri net)** A Petri net [14] is a 5-tuple:

$$PN = (P, T, F, W, M_0)$$

where:

1. $P = \{p_1, p_2, ..., p_m\}$ is a finite set of places;
2. $T = \{t_1, t_2, ..., t_n\}$ is a finite set of transitions;
3. $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation);
4. $W : F \rightarrow \{1, 2, 3, ...\}$ is a weight function;
5. $M_0 : P \rightarrow \{0, 1, 2, 3, ...\}$ is the initial marking;

This class of Petri net has two kinds of nodes, called places ($P$) represented by circles and transitions ($T$) represented by bars, such that $P \cap T =$ and $P \cup T \neq$ . Figure 1 depicts the basic elements of a simple PN. The set of arcs $F$ is used to denote the places connected to a transition (and vice-versa). $W$ is a weight function for the set of arcs. In this case, each arc is said to have multiplicity k, where k represents the respective weight of the arc. Figure 2 shows multiple arcs connecting places and transitions in a compact way by a single arc labeling it with its weight or multiplicity k.

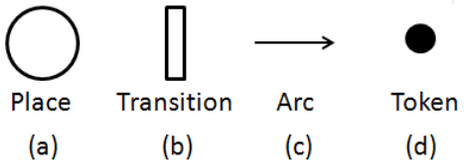


**Figure 1.** Petri net basic elements.

Places and transitions may have several interpretations. Using the concept of conditions and events, places represent conditions, and transitions represent events, such that, an event may have several pre-conditions and post-conditions. For more interpretations, Table 1 shows other meanings for places and transitions [14].

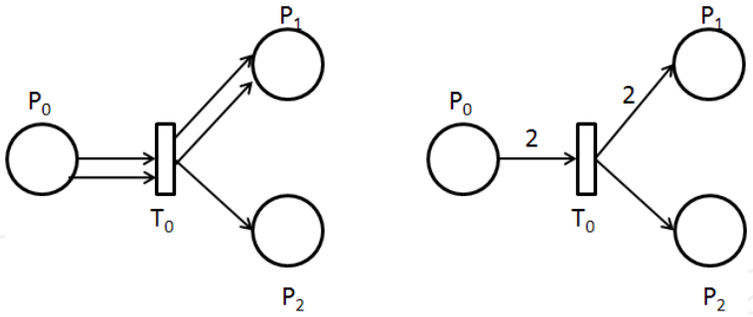


**Figure 2.** Compact representation of a PN

| Input Places | Transitions | Output Places |
|---|---|---|
| pre-conditions | events | post-conditions |
| input data | computation step | output data |
| input signals | signal processor | output signals |
| resource needed | tasks | resource releasing |
| conditions | logical clauses | conclusions |
| buffers | processor | buffers |

**Table 1.** Interpretation for places and transitions.

It is important to show that there are another way to represent PN's elements. As an example, the set of input and output places of transitions is shown in Definition 2.2. Similarly, the set of input and output transitions of determinate place is shown in Definition 2.3.

**Definition 2.2. (Input and Output Transitions of a place)** The set of input transitions (also called pre-set) of a place $p_i \in P$ is:

$$\text{label} = \bullet p_i = \{t_j \in T | (t_j, p_i) \in F\}.$$

and the set of output transitions (also called post-set) is:

$$\text{label} = p_i \bullet = \{t_j \in T | (p_i, t_j) \in F\}.$$

**Definition 2.3. (Input and output places of a transition)** The set of input places of a transition $t_j \in T$ is:

$$\text{label} = \bullet t_j = \{p_i \in P | (p_i, t_j) \in F\}.$$

and the set of output places of a transition $t_j \in T$ is:

$$\text{label} = t_j \bullet = \{p_i \in P | (t_j, p_i) \in F\}.$$

*2.2.1. Marked Petri nets*

A marking (also named token) has a primitive concept in PNs such as place and transitions. Markings are information attributed to places; the number and mark distributions consist of the net state in determined moment. The formal definitions are presented as follows.

**Definition 2.4. (Marking)** Considering the set of places $P$ in a net $N$, the formal definition of marking is represented by a function that maps the set of places P into non negative integers $M : P \rightarrow \mathbb{N}$.

**Definition 2.5. (Marking vector)** Considering the set of places $P$ in a net $N$, the marking can be defined as a vector $M = (M(p_1), ..., M(p_n))$, where $n = \#(P)$, $\forall p_i \in P \ / \ M(p_i) \in \mathbb{N}$. Thus, such vector gives the number of tokens in each place for the marking $M_i$.

**Definition 2.6. (Marked net)** A marked Petri net is defined by a tupla $NM = (N; M_0)$, where $N$ is the net structure and $M_0$ is the initial marking.

A marked Petri net contains tokens, which reside in places, travel along arcs, and their flow through the net is regulated by transitions. A peculiar distribution (M) of the tokens in the places, represents a specific state of the system. These tokens are denoted by black dots inside the places as shown in Figure 1 (d).

*2.2.2. Transition enabling and firing*

The behavior of many systems can be described in terms of system states and their changes. In order to simulate the dynamic behavior of a system, a state (or marking) in a Petri net is changed according to the following firing rule:

1. A transition $t$ is said to be enabled, if each input place $p$ of $t$ is marked with at least the number of tokens equal to the multiplicity of its arc connecting p with $t$. Adopting a mathematical notation, an enabled transition $t$ for given marking $m_i$ is denoted by $m_i[t >$, if $m_i(p_j) \geq W(p_j, t), \forall p_j \in P$.
2. An enabled transition may or may not fire (depending on whether or not the respective event takes place).
3. The firing of an enabled transition $t$ removes tokens (equal to the multiplicity of the input arc) from each input place $p$, and adds tokens (equal to the multiplicity of the output arc) to each output place $p'$. Using a mathematical notation, the firing of a transition is represented by the equation $m_j(p) = m_i(p) - W(p, t) + W(t, p), \forall p \in P$. If a marking $m_j$ is reachable from $m_i$ by firing a transition $t$, it is denoted by $m_i[t > m_j$ .

Figure 3 (a) shows the mathematical representation of a Petri net model with three places ($p_0$, $p_1$, $p_2$) and one transition ($t_0$). Besides, there is one arc connecting the place $p_0$ to the transition $t_0$ with weight two, one arc from the place $p_1$ to the transition $t_0$ with weight one, and one arc connecting the transition $t_0$ to the place $p_2$ with weight two. The initial marking ($m_0$) is represented by three tokens in the place $p_0$ and one token in the place $p_1$. Figure 3 (b)

outlines its respective graphical representation, and Figure 3 (c) provides the same graphical representation after the firing of $t_0$. For this example, the set of reachable markings is m = $\{m_0 = (3,1,0), m_1 = (1,0,2)\}$. The marking $m_1$ was obtained by firing $t_0$, such that, $m_1(p_0)$ = 3 - 2 + 0, $m_1(p_1)$ = 1 - 1 + 0, and $m_1(p_2)$ = 0 - 0 + 2.

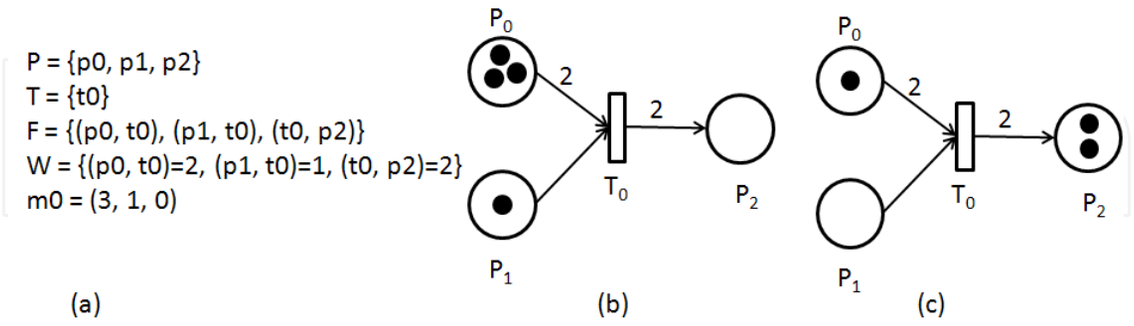


**Figure 3.** (a) Mathematical formalism; (b) Graphical representation before the firing of $t_0$; (c) Graphical representation after the firing of $t_0$.

There are two particular cases which the firing rule happens differently. The first one is a transition without any input place that is called as a *source* transition, and the other one is a transition without any output place, named *sink* transition. A *source* transition is unconditionally enabled, and the firing of a *sink* transition consumes tokens, but does not produce any. Figure 4 (a) shows a *source* transition, and Figure (b) 4 depicts a *sink* transition. In both, the markings are represented before and after their respective firing.

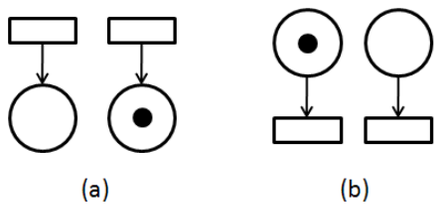


**Figure 4.** (a) *Source* transitions; (b) *Sink* transitions.

**Definition 2.7. (Source transitions)** A transition is said to be source if, and only if, $I(p,t) = 0$, $\forall p \in P$.

**Definition 2.8. (Sink transitions)** A transition is said to be sink if, and only if, $O(p,t) = 0$, $\forall p \in P$.

**Definition 2.9. (Inhibitor arc)** Originally not present in PN, the introduction of the concept of inhibitor arc increases the modeling power of PN, adding the ability of testing if a place does not have tokens. In the presence of an inhibitor arc, a transition is enabled to fire if each input place connected by a normal arc has a number of tokens equal to the arc weight, and if each input place connected by an inhibitor arc has no tokens. Figure 5 illustrates an inhibitor arc connecting the input place $p_0$ to the transition $t_0$, which is denoted by an arc finished with a small circle. In such Figure, the transition $t_0$ is enabled to fire.

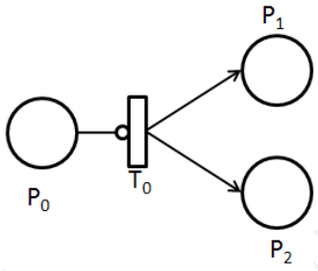


**Figure 5.** PN with an inhibitor arc.

**Definition 2.10. (Pure net)** A Petri net is said to be pure if it has no self-loops. A pair of a place $p$ and transition $t$ is called a self-loop if $p$ is both an input and output place of $t$. Figure 6 shows a self-loop net.

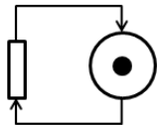

**Figure 6.** Self-Loop.

## 2.3. Elementary structures

Elementary nets are used as building blocks in the specification of more complex applications. Figure 7 shows five structures, namely, (a) sequence, (b) fork, (c) synchronization, (d) choice, and (e) merging.

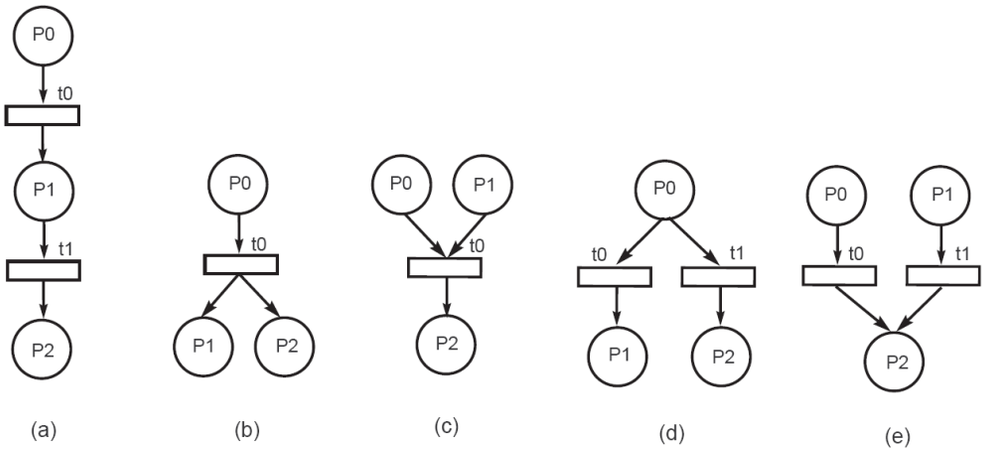


**Figure 7.** Elementary PN Structures.

### Sequence

Sequence structure represents sequential execution of actions, provided that a condition is satisfied. After the firing of a transition, another transition is enabled to fire. Figure 7(a) depicts an example of this structure in which a mark in place $p_0$ enables the transition $t_0$. The firing of transition $t_0$ enables the transition $t_1$ ($p_1$ is marked).

**Fork**

Figure 7(b) shows an example of a fork structure that allows the creation of parallel processes.

**Join**

Generally, concurrent activities need to synchronize with each other. This net (Figure 7(c)) combines two or more nets, allowing that another process continues this execution only after the end of predecessor processes.

**Choice**

Figure 7(d) depicts a choice model, in which the firing of the transition $t0$ disables the transition $t_1$. This building block is suited for modeling if-then-else statement, for instance.

**Merging**

The merging is an elementary net that allows the enabling of the same transition by two or more processes. Figure 7(e) shows a net with two independent transitions ($t_0$ and $t_1$) that have an output place in common ($P2$). Therefore, firing of any of these two transitions, a condition is created ($p_2$ is marked) which allows the firing of another transition (not shown in the figure).

**Confusions**

The mixing between conflict and concurrency is called confusion. While conflict is a local phenomenon in the sense that only the pre-sets of the transitions with common input places are involved, confusion involves firing sequences. Figure 8 depicts two types of confusions: (a) symmetric confusion, where two transitions $t_1$ and $t_3$ are concurrent while each one is in conflict with transition $t_2$; and (b) asymmetric confusion, where $t_1$ is concurrent with $t_2$, but will be in conflict with $t_3$ if $t_2$ fires first.

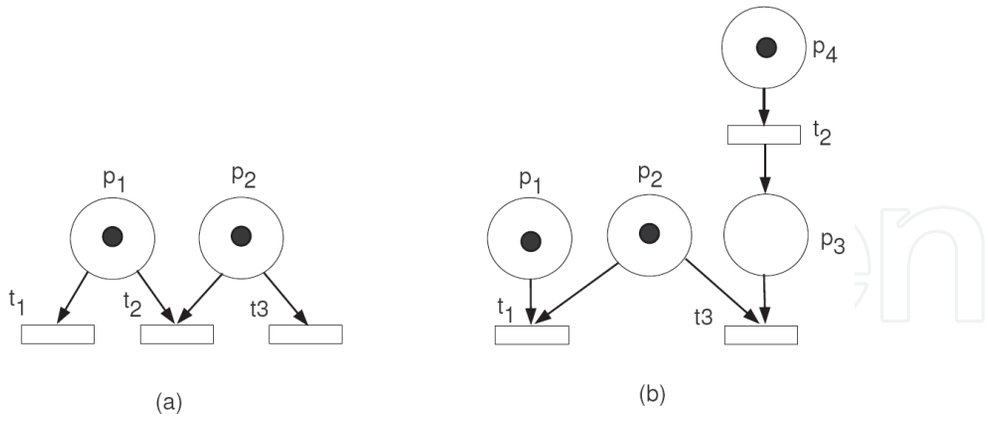


**Figure 8.** (a) symmetric confusion; (b) asymmetric confusion.

## 2.4. Petri nets modeling examples

In this section, several simple examples are given in order to introduce how to model some basic concepts such as parallel process and mutual exclusion in Petri nets.

### Parallel processes

In order to represent parallel processes, a model may be obtained by composing the model for each individual process with a fork and synchronization models. Two transitions are said to be parallel (or concurrent), if they are causally independent, i.e., one transition may fire either before (or after) or in parallel with the other.

Figure 9 depicts an example of parallel process, where transitions $t_1$ and $t_2$ represent parallel activities. When transition $t_0$ fires, it creates marks in both output places ($p_0$ and $p_1$), representing a concurrency. When $t_1$ and $t_2$ are enabled for firing, each one may fire independently. The firing of $t_3$ depends on two pre-conditions, $p_2$ and $p_3$, implying that the system only continues if $t_1$ and $t_2$ have been fired.

Figure 9 presents a net in which each place has exactly one incoming arc and exactly one outgoing arc. Thus, such model represents a sub-class of Petri nets known as marked graphs. Marked graphs allow representation of concurrency but not decisions or conflicts.

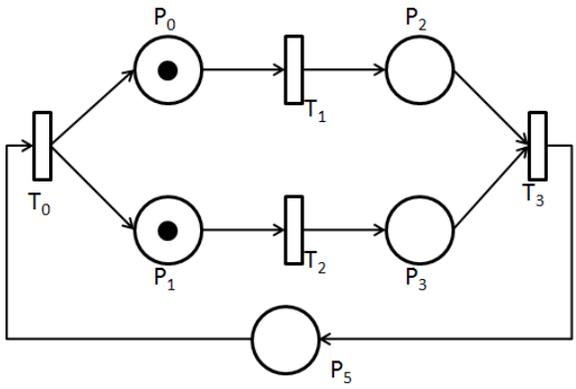


**Figure 9.** A Petri net representing parallel activities.

### Mutual exclusion

The sharing of resources and/or data are common in many system applications, in which most of resources and data should be accessed in a mutual exclusive way. Resources (or data variable) may be modeled by a place with tokens representing the amount of resources. This place is seen as pre-conditions for all transitions that need such resource. After the use of one resource, it must be released. Figure 10 depicts an example of a machine that is accessed in a mutual exclusive way.

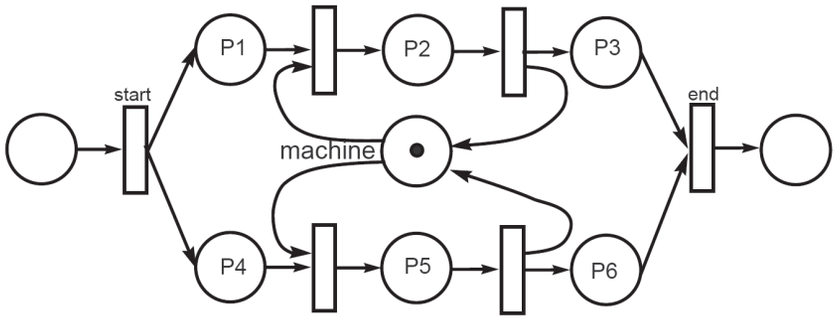


**Figure 10.** Mutual Exclusion.

**Dataflow computation**

Petri nets can be used to represent not only the control-flow but also the data-flow. The net shown in Figure 11 is a Petri net representation of a dataflow computation. A dataflow is characterized by the concurrent instruction execution (or transitions firing) as soon as the operands (pre-conditions) are available. In the Petri net representation, tokens may denote values of current data as well as the availability of data. The instructions are represented by transitions such as *Add* and *Subtract* that can be executed in parallel. After that, if the activity *Subtract* has computed a result different from zero, meaning that the pre-conditions to perform *divide* operation were satisfied. Afterwards, when the transition *divide* occur, the dataflow computation is completed.

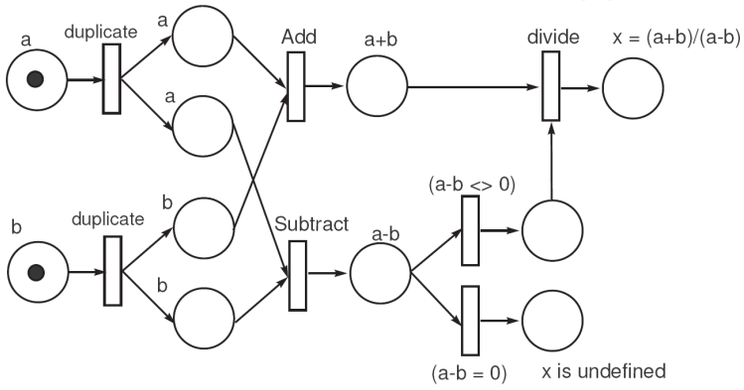


**Figure 11.** Dataflow example.

## 2.5. Petri nets properties

The PN properties allow a detailed analysis of the modeled system. For this, two types of properties have been considered in a Petri net model: behavioral and structural properties. Behavioral properties are those which depend on the initial marking. Structural properties, on the other hand, are those that are marking-independent.

### *2.5.1. Behavioral properties*

This section, based on [14], describes some behavioral properties, since such properties are very important when analyzing a given system.

**Reachability**

The firing of an enabled transition changes the token marking in a Petri net, and a sequence of firings results in a sequence of markings. A marking $M_n$ is said to be reachable from a marking $M_0$ if there exists a sequence of firings that transforms $M_0$ to $M_n$.

A firing (or occurrence) sequence is denoted by $\sigma = t_1, t_2, ..., t_n$. In this case, $m_i$ is reachable from $m_0$ by $\sigma$, and it is denoted by $m_0[\sigma > m_i$. The set of all possible reachable markings from $m_0$ in a net (PN, $m_0$) is denoted by R(PN,$m_0$), or simply R($m_0$). The set of all possible firing sequence from $m_0$ in a net (PN,$m_0$) is denoted by L(PN, $m_0$), or simply L($m_0$).

**Boundedness**

A Petri net is said to be bounded if the number of tokens in each place does not exceed a finite number k for any marking reachable from $M_0$. In a formal way, $M(p) \leq k$, $\forall p \in P$ and $\forall M \in R(M_0)$.

**Safe**

When the number of tokens in each place does not exceed the number "1" (one), such Petri net is said to be safe. It is important to state that if a net is bounded or safe, it is guaranteed that there will be no overflows in any place, no matter the firing sequence adopted.

**Deadlock freedom**

A PN is said to be deadlock free if there is no reachable marking such that no transition is enabled.

**Liveness**

In an informal way, a Petri net is said to be live if it is guaranteed that no matter what firing sequence is chosen, it continues in deadlock-free operation. The formal definition, a Petri net (N,$M_0$) is said to be live if, no matter what marking has been reached from $M_0$, it is possible to ultimately fire any transition of the net.

Liveness is an ideal property for many real systems. However, it is very strong and too costly to verify. Thus, the liveness condition is relaxed in different levels. A transition $t$ is said to be live at the following levels:

- $L_0$ Live (dead), if $t$ can never be fired in any firing sequence in L($m_0$), it is a dead transition.
- $L_1$-Live (potentially firable), if it can be fired at least once in some firing sequence in L($m_0$).
- $L_2$-Live if, given any positive integer k, $t$ can be fired at least k times in some firing sequence in L($m_0$).
- $L_3$-Live if there is an infinite-length firing sequence in L($m_0$) in which $t$ is fired infinitely.
- $L_4$-Live (or simply live), if it is L1-Live for every marking m in R($m_0$).

**Persistence**

A Petri net is said to be persistent if, for any two enabled transitions, the firing of one transition will not disable the other. Once a transition is enabled in a persistent net, it is continue to be enabled until it fires. Persistency is closed related to conflict-free nets. It is worth noting that all marked graph are persistent, but not all persistent nets are marked graphs. Persistence is a very important property when dealing with parallel system design and speed-independent asynchronous circuits.

*2.5.2. Structural properties*

**Structural liveness**

A PN $N$ is said to be structurally live if there is a live initial marking for $N$.

**Structural boundedness**

A PN $N$ is said to be structurally bounded if it is bounded for any finite initial marking $M_0$.

**Structural conservativeness**

A PN that provides a constant weighted sum of tokens for any reachable marking when considering any initial marking is said to be structural conservative.

**Structural repetitiveness**

A PN is classified as repetitive if there is an initial marking $m_0$ and an enabled firing sequence from $m_0$ such that every transition of the net is infinitely fired. On the other hand, if only some of these transitions are fired infinitely often in the sequence $\sigma$, this net is called partially repetitive.

**Consistence**

A net is classified as consistent if there is an initial marking $m_0$ and an enabled firing sequence from $m_0$ back to $m_0$ such that every transition of the net is fired at least once. If only some of these transitions are not fired in the sequence $\sigma$, this net is called partially consistent.

## 2.6. Stochastic Petri nets

Petri nets [17] are a classic tool for modeling and analyzing discrete event systems which are too complex to be described by automata or queueing models. Time (stochastic delays) and probabilistic choices are essential aspects for a performance evaluation model. We adopt the usual association of delays and weights with transitions [11] in this paper, and adopt the extended stochastic Petri net definition similar to [9]:

Let $SPN = (P, T, I, O, H, \Pi, G, M_0, Atts)$ be a stochastic Petri net, where

- $P = \{p_1, p_2, ..., p_n\}$ is the set of places, which may contain tokens and form the discrete state variables of a Petri net.
- $T = \{t_1, t_2, ..., t_m\}$ is the set of transitions, which model active components.
- $I \in (\mathbb{N}^n \to \mathbb{N})^{n \times m}$ is a matrix of marking-dependent multiplicities of input arcs, where $i_{jk}$ entry of $I$ gives the (possibly marking-dependent) arc multiplicity of input arcs from place $p_j$ to transition $t_k$ [$A \subseteq (P \times T) \cup (T \times P)$ — set of arcs]. A transition is only enabled if there are enough tokens in all input places.
- $O \in (\mathbb{N}^n \to \mathbb{N})^{n \times m}$ is a matrix of marking dependent multiplicities of output arcs, where $o_{jk}$ entry of $O$ specifies the possibly marking-dependent arc multiplicity of output arcs from transition $t_j$ to place $p_k$. When a transition fires, it removes the number of tokens specified by the input arcs from input places, and adds the amount of tokens given by the output arcs to all output places.
- $H \in (\mathbb{N}^n \to \mathbb{N})^{n \times m}$ is a matrix of marking-dependent multiplicities describing the inhibitor arcs, where $h_{jk}$ entry of $H$ returns the possibly marking-dependent arc multiplicity of an inhibitor arc from place $p_j$to transition $t_k$. In the presence of an inhibitor arc, a transition is enabled to fire only if every place connected by an inhibitor arc contains fewer tokens than the multiplicity of the arc.
- $\Pi \in \mathbb{N}^m$ is a vector that assigns a priority level to each transition. Whenever there are several transitions fireable at one point in time, the one with the highest priority fires first and leads to a state change.

- $M_0 \in \mathbb{N}^n$ is a vector that contains the initial marking for each place (initial state).
- $Atts : (Dist, W, G, Policy, Concurrency)^m$ comprises a set of attributes for the $m$ transitions, where
  - $Dist \in \mathbb{N}^m \rightarrow \mathcal{F}$ is a possibly marking dependent firing probability distribution function. In a stochastic timed Petri net, time has to elapse between the enabling and firing of a transition. The actual firing time is a random variable, for which the distribution is specified by $\mathcal{F}$. We differ between immediate transitions ($\mathcal{F} = 0$) and timed transitions, for which the domain of $\mathcal{F}$ is $(0, \infty)$.
  - $W \in \mathbb{R}^+$ is the weight function, that represents a firing weight $w_t$ for immediate transitions or a rate $\lambda_t$ for timed transitions. The latter is only meaningful for the standard case of timed transitions with exponentially distributed firing delays. For immediate transitions, the value specifies a relative probability to fire the transition when there are several immediate transitions enabled in a marking, and all have the same probability. A random choice is then applied using the probabilites $w_t$.
  - $G \in \mathbb{N}^n \rightarrow \{true, false\}$ is a function that assigns a guard condition related to place markings to each transition. Depending on the current marking, transitions may not fire (they are disabled) when the guard function returns false. This is an extension of inhibitor arcs.
  - $Policy \in \{prd, prs\}$ is the preemption policy (*prd — preemptive repeat different* means that when a preempted transition becomes enabled again the previously elapsed firing time is lost; *prs — preemptive resume*, in which the firing time related to a preempted transition is resumed when the transition becomes enabled again),
  - $Concurrency \in \{ss, is\}$ is the concurrency degree of transitions, where *ss* represents single server semantics and *is* depicts infinity server semantics in the same sense as in queueing models. Transitions with policy *is* can be understood as having an individual transition for each set of input tokens, all running in parallel.

In many circumstances, it might be suitable to represent the initial marking as a mapping from the set of places to natural numbers ($m_0 : P \rightarrow \mathbb{N}$), where $m_0(p_i)$ denotes the initial marking of place $p_i$. $m(p_i)$ denotes a reachable marking (reachable state) of place $p_i$. In this work, the notation $\#p_i$ has also been adopted for representing $m(p_i)$.

## 2.7. Dependability

Dependability of a computer system must be understood as the ability to deliver services with respect to some agreed-upon specifications of desired service that can be fully trusted [1, 13]. Indeed, dependability is related to disciplines such as fault tolerance and reliability. Reliability is the probability that the system will deliver a set of services for a given period of time, whereas a system is fault tolerant when it does not fail even when there are faulty components. Availability is also another important concept, which quantifies the mixed effect of both failure and repair process in a system. In general, availability and reliability are related concepts, but they differ in the sense that the former may consider maintenance of failed components [8] (e.g., a failed component is restored to a specified condition).

In many situations, modeling is the method of choice either because the system might not yet exist or due to the inherent complexity for creating specific scenarios under which the system should be evaluated. In a very broad sense, models for dependability evaluation

can be classified as simulation and mathematical models. However, this does not mean that mathematical models cannot be simulated. Indeed, many mathematical models, besides being analytically tractable, may also be evaluated by simulation. Mathematical models can be characterized as being either state-based or non-state-based.

Dependability metrics (e.g., availability, reliability and downtime) might be calculated either by using RBD or SPN (to mention only the models adopted in this work). RBDs allow to one represent component networks and provide closed-form equations, so the results are usually obtained faster than using SPN simulation. Nevertheless, when faced with representing maintenance policies and redundant mechanisms, particularly those based on dynamic redundancy methods, such models experience drawbacks concerning the thorough handling of failures and repairing dependencies. On the other hand, state-based methods can easily consider those dependencies, so allowing the representation of complex redundant mechanisms as well as sophisticated maintenance policies. However, they suffer from the state-space explosion. Some of those formalism allow both numerical analysis and stochastic simulation, and SPN is one of the most prominent models of such class.

If one is interested in calculating the availability ($A$) of given device or system, he/she might need either the uptime and downtime or the time to failure ($TTF$) and time to repair ($TTR$). Considering that the uptime and downtime are not available, the later option is the mean. If the evaluator needs only the mean value, the metrics commonly adopted are Mean Time to Failure ($MTTF$) and Mean Time To Repair ($MTTR$) (other central values might also be adopted). However, if one is also interested in the availability variation, the standard deviation of time to failure ($sd(TTF)$), and the respective standard deviation of time to repair ($sd(TTR)$) allow one the estimate the availability variation.

The availability ($A$) is obtained by steady-state analysis or simulation, and the following equation expresses the relation concerning $MTTF$ and $MTTR$:

$$A = \frac{MTTF}{MTTF + MTTR} \tag{1}$$

Through transient analysis or simulation, the reliability ($R$) is obtained, and, then, the $MTTF$ can be calculated as well as the standard deviation of the Time To Failure ($TTF$):

$$MTTF = \int_0^\infty t f(t) dt = \int_0^\infty -\frac{dR(t)}{dt} t dt = \int_0^\infty R(t) dt \tag{2}$$

$$sd(TTF) = \sqrt{\int_0^\infty t^2 f(t) dt - (MTTF)^2} \tag{3}$$

Considering a given period $t$, $R(t)$ is the probability that the time to failure is greater than or equal to t. Regarding exponential failure distributions, reliability is computed as follows:

$$R(t) = exp\left[-\int_0^t \lambda(t')dt'\right] \tag{4}$$

where $\lambda(t')$ is the instantaneous failure rate.

One should bear in mind that, for computing reliability of a given system service, the repairing activity of the respective service must not be represented. Besides, taking into account $UA = 1 - A$ (unavailability) and Equation 1, the following equation is derived

$$MTTR = MTTF \times \frac{UA}{A} \tag{5}$$

As well, the standard deviation of the Time To Repair ($TTR$) can be calculated as follows:

$$sd(TTR) = sd(TTF) \times \frac{UA}{A} \tag{6}$$

Next, $\frac{MTTF}{sd(TTF)}$ (and $\frac{MTTR}{sd(TTR)}$) are computed for choosing the expolinomial distribution that best fits the $TTF$ and $TTR$ distributions [6, 22].

Figure 12 depicts the generic simple component model using SPN, which provides a high-level representation of a subsystem. One should notice the trapezoidal shape of transitions (high-level transition named s-transition). This shape means that the time distributions of such transitions are not exponentially distributed, instead they should be refined by subnets. The delay assigned to s-transition $f$ is the $TTF$ and the delay of s-transition $r$ is the $TTR$. If the $TTF$ and $TTR$ are exponentially distributed, the shape of the transitions should be the regular one (white rectangles) and $TTF$ and $TTR$ should be summarized by the respective $MTTF$ and $MTTR$.

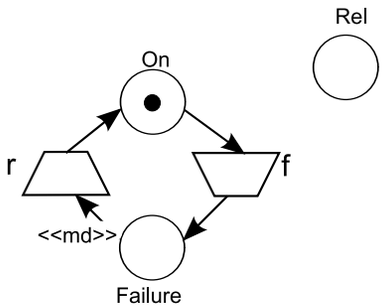


**Figure 12.** Generic simple model - SPN

A well-established method that considers *expolynomial distribution* random variables is based on distribution moment matching. The moment matching process presented in [6] takes into account that Hypoexponential and Erlangian distributions have the average delay ($\mu$) greater than the standard-deviation ($\sigma$) -$\mu > \sigma$-, and Hyperexponential distributions have $\mu<\sigma$, in order to represent an activity with a generally distributed delay as an Erlangian or a Hyperexponential subnet referred to as s-transition[1]. One should note that in cases where these distributions have $\mu = \sigma$, they are, indeed, equivalent to an exponential distribution with parameter equal to $\frac{1}{\mu}$. Therefore, according to the coefficient of variation associated with an activity's delay, an appropriate s-transition implementation model could be chosen. For each s-transition implementation model (see Figure 13), a set of parameters should be configured for matching their first and second moments. In other words, an associated delay distribution (it might have been obtained by a measuring process) of the

[1] In this work, $\mu$ could be $MTTF$ or $MTTR$ and the $\sigma$ could represent $sd(TTF)$ or $sd(TTR)$, for instance.

original activity is matched with the first and second moments of s-transition (*expolynomial distribution*). According to the aforementioned method, one activity with $\mu<\sigma$ is approximated by a two-phase Hyperexponential distribution with parameters

$$r_1 = \frac{2\mu^2}{(\mu^2+\sigma^2)}, \tag{7}$$

$$r_2 = 1 - r_1 \tag{8}$$

and

$$\lambda = \frac{2\mu}{(\mu^2+\sigma^2)}. \tag{9}$$

where $\lambda$ is the rate associated to phase 1, $r_1$ is the probability of related to this phase, and $r_2$ is the probability assigned to phase 2. In this particular model, the rate assigned to phase 2 is assumed to be infinity, that is, the related average delay is zero.

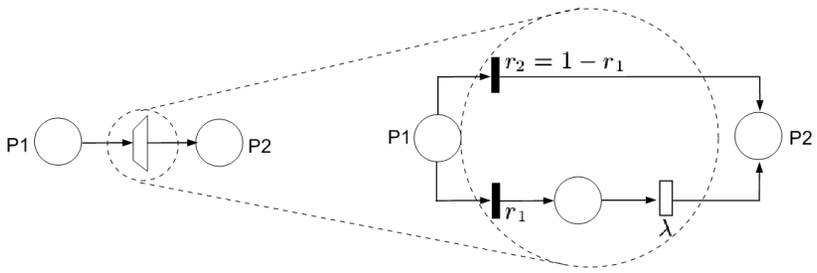


**Figure 13.** Hyperexponential Model

Activities with coefficients of variation less than one might be mapped either to Hypoexponential or Erlangian s-transi-
tions. If $\frac{\mu}{\sigma} \notin \mathbb{N}$, $\frac{\mu}{\sigma} \neq 1, (\mu, \sigma \neq 0)$, the respective activity is represented by a Hypoexponential distribution with parameters $\lambda_1$, $\lambda_2$(exponential rates); and $\gamma$, the integer representing the number of phases with rate equal to $\lambda_2$, whereas the number of phases with rate equal to $\lambda_1$ is one. In other words, the s-transition is represented by a subnet composed of two exponential and one immediate transitions. The average delay assigned to the exponential transition $t_1$ is equal to $\mu_1$ ($\lambda_1 = 1/\mu_1$), and the respective average delay assigned to the exponential transition $t_2$ is $\mu_2(\lambda_2 = 1/\mu_2)$. $\gamma$ is the integer value considered as the weight assigned to the output arc of transition $t_1$ as well as the input arc weight value of the immediate transition $t_3$ (see Figure 14). These parameters are calculated by the following expressions:

$$(\frac{\mu}{\sigma})^2 - 1 \leq \gamma < (\frac{\mu}{\sigma})^2, \tag{10}$$

$$\lambda_1 = \frac{1}{\mu_1} \; and \; _2 = \frac{1}{\mu_2}, \tag{11}$$

where

$$\mu_1 = \frac{\mu \pm \sqrt{\gamma(\gamma+1)\sigma^2 - \gamma\mu^2}}{\gamma+1}, \tag{12}$$

$$\mu_2 = \frac{\gamma\mu \mp \sqrt{\gamma(\gamma+1)\sigma^2 - \gamma\mu^2}}{\gamma+1} \tag{13}$$

If $\frac{\mu}{\sigma} \in \mathbb{N}, \frac{\mu}{\sigma} \neq 1, (\mu, \sigma \neq 0)$, an Erlangian s-transition with two parameters, $\gamma = (\frac{\mu}{\sigma})^2$ is an integer representing the number of phases of this distribution; and $\mu_1 = \mu/\gamma$, where $\mu_1(1/\lambda_1)$ is the average delay value of each phase. The Erlangian model is a particular case of a Hypoexponential model, in which each individual phase rate has the same value.

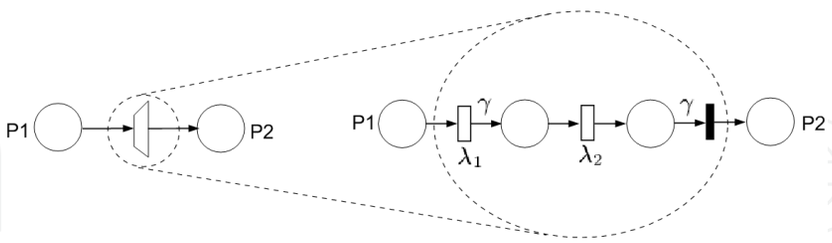


**Figure 14.** Hypoexponential Model

The reader should refer to [6] for details regarding the representation of expolinomial distributions using SPN. For the sake of simplicity, the SPN models presented in the next sections consider only exponential distributions.

Depending on the system characteristics, a RBD model (Figure 15) could be adopted instead of the SPN counterpart, whenever the former is more suitable.

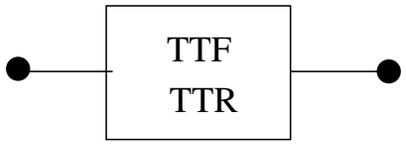


**Figure 15.** Generic simple model - RBD

## 3. Related works

In the last few years, some works have been developed to perform dependability analysis of data center systems [24][26][27]. Reliability (which encompasses both the durability of the data and its availability for access) correspond to the primary property that data center users desire [2], .

Robidoux [28] proposes Dynamic RBD (DRBD) model, an extension to RBD, which supports reliability analysis of systems with dependence relationships. The additional blocks (in relation to RBD) to model dependence, turned the DRBD model complex. The DRBD model is automatic converted to CPN model in order to perform behavior properties analysis which may certify the correctness of the model [18]. It seems that an interesting alternative would be to model the system directly using CPN or any other formalism (e.g., SPN) which is able to perform dependability analysis as well as to model dependencies between components.

Wei [25] presents an hierarchical method to model and analyze virtual data center (VDC). The approach combines the advances of both RBD and General SPN (GSPN) for quantifying availability and reliability. Data center power architectures are not the focus of their research and the proposed models are specific for modeling VDC.

Additionally, redundancies on components to increase system reliability are costly. [7] propose an approach for reliability evaluation and risk analysis of dynamic process systems using stochastic Petri nets.

Different from previous works, this paper proposes a set of models to the quantification of dependability metrics in the context of data center design. Furthermore, the adopted methodology for the quantification of those values takes into account a hybrid modeling approach, which utilizes RBD and SPN whenever they are best suited. The idea of mixing state (SPN) and non-state (RBD) based models is not new (e.g., [23]), but, as far as we are concerned, there is no similar work that applies such technique on the evaluation of data center infrastructures. Besides, a tool is proposed to automate several activities.

## 4. Dependability models

The following sections presents the adopted dependability models.

**RBD Models**

Reliability Block Diagram (RBD) [8] is a combinatorial model that was initially proposed as a technique for calculating reliability of systems using intuitive block diagrams. Such a technique has also been extended to calculate other dependability metrics, such as availability and maintainability [10]. Figure 16 depicts two examples, in which independent blocks are arranged through series (Figure 16(a)) and parallel (Figure 16(b)) compositions.

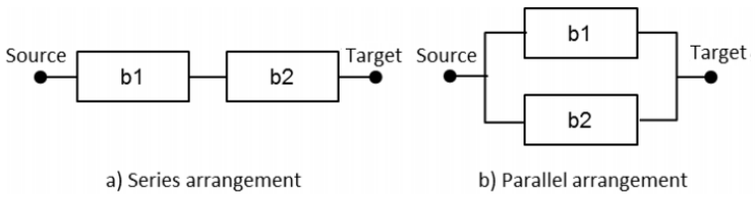


**Figure 16.** Reliability Block Diagram

In the series arrangement, if a single component fails, the whole system is no longer operational. Assuming a system with $n$ independent components, the reliability (instantaneous availability or steady state availability) is obtained by

$$P_s = \prod_{i=1}^{n} P_i \tag{14}$$

where $P_i$ is the reliability - $R_i(t)$ (instantaneous availability ($A_i(t)$) or steady state availability ($A_i$)) of block $b_i$.

For a parallel arrangement (see Figure 16(b)), if a single component is operational, the whole system is also operational. Assuming a system with $n$ independent components, the reliability (instantaneous availability or steady state availability) is obtained by

$$P_p = 1 - \prod_{i=1}^{n} (1 - P_i) \tag{15}$$

where $P_i$ is the reliability - $R_i(t)$ (instantaneous availability ($A_i(t)$) or steady state availability ($A_i$)) of block $b_i$.

A k-out-of-n system functions if and only if $k$ or more of its $n$ components are functioning. Let $p$ be the success probability of each of those blocks. The system success probability (reliability or availability) is depicted by:

$$\Sigma_{i=k}^{n} \binom{n}{b} p^k (1-p)^{n-k} \tag{16}$$

For other examples and closed-form equations, the reader should refer to [10].

**SPN Models**
This section presents two proposed SPN building block for obtaining dependability metrics.

**Simple Component.** The simple component has two states: functioning or failed. To compute its availability, $MTTF$ and $MTTR$ should be represented. Figure 17 shows the SPN model of the "simple component", which has two parameters (not depicted in the figure), namely $X\_MTTF$ and $X\_MTTR$, representing the delays associated to the transitions $X\_Failure$ and $X\_Repair$, respectively.

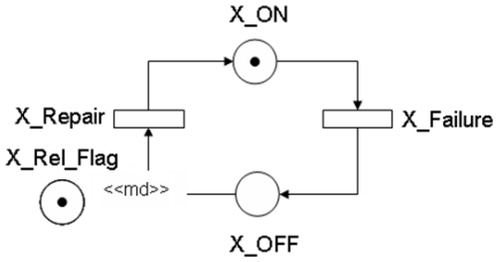


**Figure 17.** Simple component model

Places $X\_ON$ and $X\_OFF$ are the model component's activity and inactivity states, respectively. The simple component also includes an arc from $X\_OFF$ to X_Repair with multiplicity depending on place marking. The multiplicity is defined through the expression IF(#$X\_Rel\_Flag = 1$):2 ELSE 1, where place $X\_Rel\_Flag$ models the evaluation of reliability/availability. Hence, if condition #$X\_Rel\_Flag = 1$ is true, then the evaluation refers to reliability. Otherwise, the evaluation concerns availability.

Besides, although simple component model has been presented using the exponential distribution, other expolinomial distributions that best fits the $TTF$ and $TTR$ may be adopted following the techniques presented in [22].

**Cold standby.** A cold standby redundant system is composed by a non-active spare module that waits to be activated when the main active module fails. Figure 18 depicts the SPN model of this system, which includes four places, namely $X\_ON$, $X\_OFF$, $X\_Spare1\_ON$, $X\_Spare1\_OFF$ that represent the operational and failure states of both the main and spare modules, respectively. The spare module (Spare1) is initially deactivated, hence no tokens are initially stored in places $X\_Spare1\_ON$ and $X\_Spare1\ \_OFF$. When the main module fails, the transition $X\_Activate\_Spare1$ is fired to activate the spare module.

Table 2 presents the attributes of each transition of the model. Once considering reliability evaluation (number of tokens (#) in the place $X\_Rel\_Flag = 1$), the $X\_Repair$, $X\_Activate\_Spare1$ and $X\_Repair\_Spare1$ transitions receive a huge number (many times larger than the associated MTTF or MTActivate) to represent the absence of repair. The MTActivate corresponds to the mean time to activate the spare module. Besides, when considering reliability, the weight of the edge that connects the place $X\_Wait\_Spare1$ and the $X\_Activate\_Spare1$ transition is two; otherwise, it is one. Both availability and reliability may be computed by the probability $P\{\#X\_\ ON = 1$ OR $\#X\_Spare1\ \_ON = 1\}$.

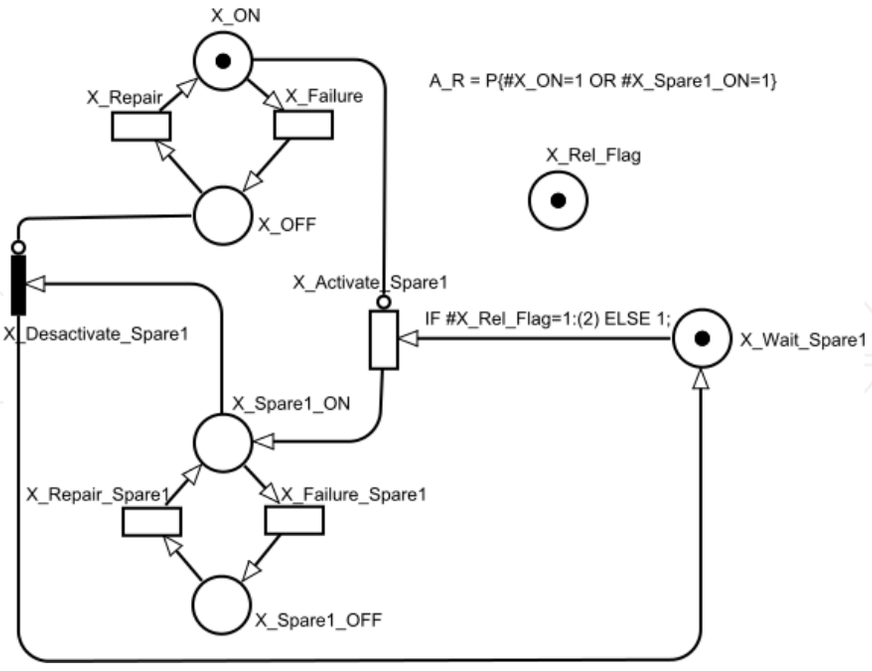


**Figure 18.** Cold standby model.

| Transition | Priority | Delay or Weight |
|---|---|---|
| X_Failure | - | X_MTTF |
| X_Repair | - | IF #X_Rel_Flag=1:($10^{13}$ x X_MTTF) ELSE X_MTTR |
| X_Activate_Spare1 | - | IF #X_Rel_Flag=1:($10^{13}$ x MTActivate) ELSE MTActivate |
| X_Failure_Spare1 | - | X_MTTF_Spare1 |
| X_Repair_Spare1 | - | IF #X_Rel_Flag=1:($10^{13}$ x X_MTTF_Spare1) ELSE X_MTTR_Spare1 |
| X_Desactivate_Spare1 | 1 | 1 |

**Table 2.** Cold standby model - Transition attributes.

## 5. Applications

This section focuses in presenting the applicability of the proposed models to perform dependability analysis of real-world data center power architectures (from HP Labs Palo Alto, U.S. [12]). The environment ASTRO was adopted to conduct the case study. ASTRO was validated through our previous work [5] [3] [4].

### 5.1. Architectures

Data center power infrastructure is responsible for providing uninterrupted, conditioned power at correct voltage and frequency to the IT equipments. Figure 19 (a) depicts a real-world power infrastructure. From the utility feed (i.e., AC Source), typically, the power goes through voltage panels, uninterruptible power supply (UPS) units, power distribution units (PDUs) (composed of transformers and electrical subpanels), junction boxes, and, finally, to rack PDUs (rack power distribution units). The power infrastructure fails (and, thus, the system) whenever both paths depicted in Figure 19 are not able to provide the power demanded (500 kW) by the IT components (50 racks). The reader should assume a path as a set of redundant interconnected components inside the power infrastructure. Another architecture is analyzed with an additional electricity generator (Figure 19 (b)) for supporting the system when both AC sources are not operational.

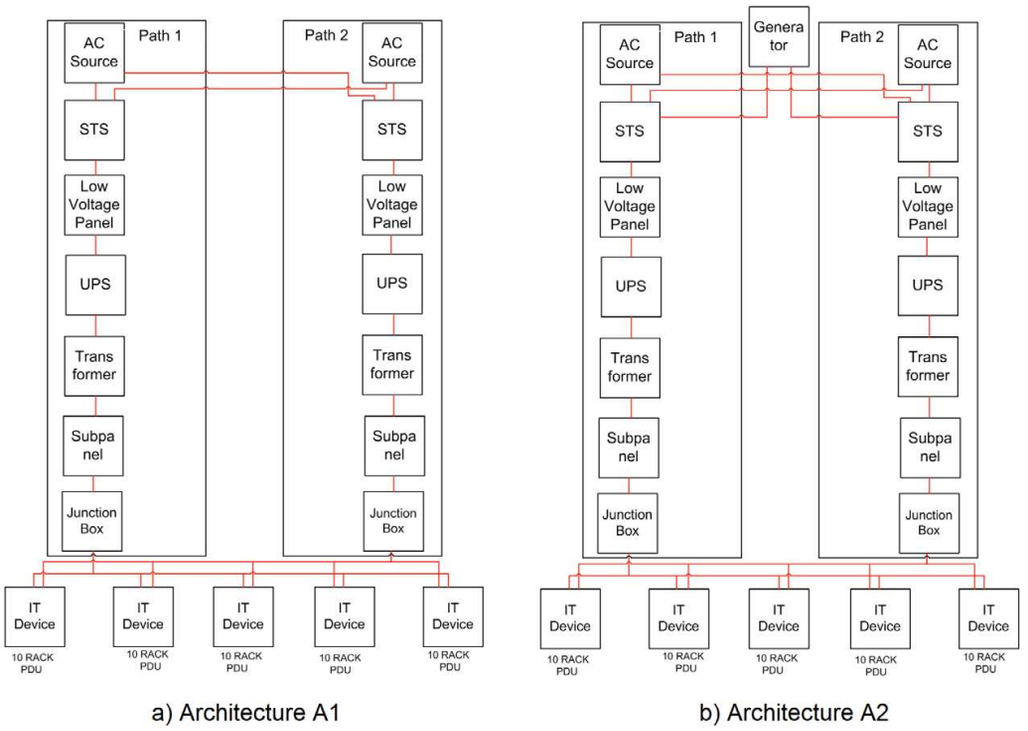


**Figure 19.** Data Center Power Architectures.

## 5.2. Models

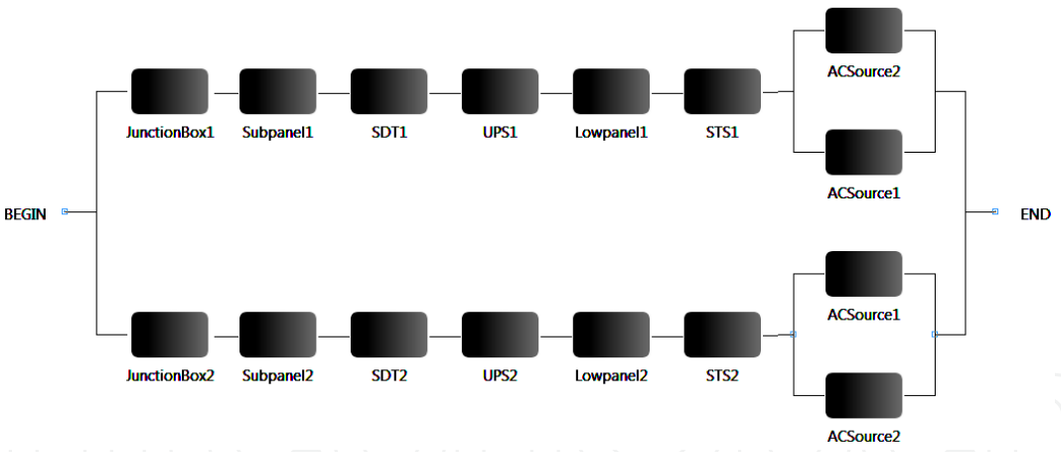


**Figure 20.** RBD of Architecture A1.

This work adopts a hierarchical methodology for conducting dependability evaluation of data center architectures. In general, the methodology aims at grouping related components in order to generate subsystem models, which are adopted to mitigate the complexity of the final system model evaluation. Thus, the final model is an approximation, but rather simpler, of a more intricate system model. One should bear in mind that the detailed model could be adopted instead, but at the expenses of complexity.

Following the adopted methodology, systems with no failure dependencies between components have been evaluated through RBD models. For instance, Figure 20 depicts the RBD model that represents the architecture A1.

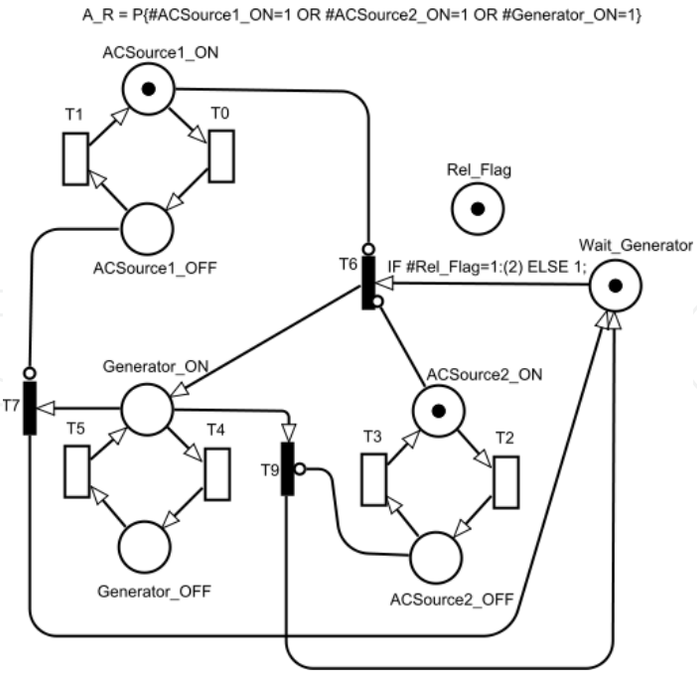


**Figure 21.** SPN of Architectures A2.

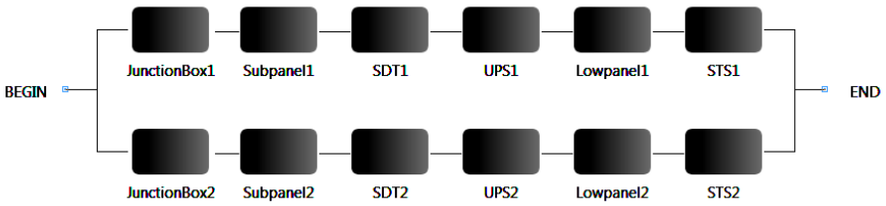


**Figure 22.** RBD of Architectures A2.

In architecture A2, the generator is only activated when both AC sources are not available. Therefore, a model that deal with dependencies must be adopted. Figure 21 shows the SPN model considering cold standby redundance to represent the subsystem composed of generator and two AC sources. Besides, we assume that UPS' batteries support the system during the generator activation. The reliability or availability is computed by the probability $P\{\#ACSource1\_ON = \text{OR } \#ACSource2\_ON = 1 \text{ OR } \#Generator\_ON = 1\}$.

The other components of the architecture A2 are modeled using RBD as shown in Figure 22. Once obtained the results of both models (RBD and the SPN model with dependencies), a RBD model with two blocks (considering the results of those models) in a serial arrangement is created. The RBD evaluation provides the dependability results of the architecture A2 system.

The adopted MTTF and MTTR values for the power devices were obtained from [21] [29] [19] and are shown in Table 3.

## 5.3. Results

Figure 23 depicts a graphical comparison between the reliability results (in number of 9's) of those two data center power architectures. The respective number of nines (-*log*[1 - A/100]) and the period of 8760 hours (1 year) are adopted. As the reader should note, the reliability of both architectures decreases when the time increases. Besides, it is also possible to notice that

| Equipment | MTTF (hs) | MTTR (hs) |
|---|---|---|
| AC Source | 4,380 | 8 |
| Generator | 2,190 | 8 |
| STS | 240,384 | 8 |
| Subpanel | 1,520,000 | 8 |
| Transformer | 1,412,908 | 8 |
| UPS | 250,000 | 8 |
| Low Voltage Panel | 1,520,000 | 8 |

**Table 3.** MTTF and MTTR values for power devices.

the generator has increased the reliability of the architecture A2. Considering the availability results, similar behavior happened. The availability has increased from 5.47 to 7.96 (in number of 9's).

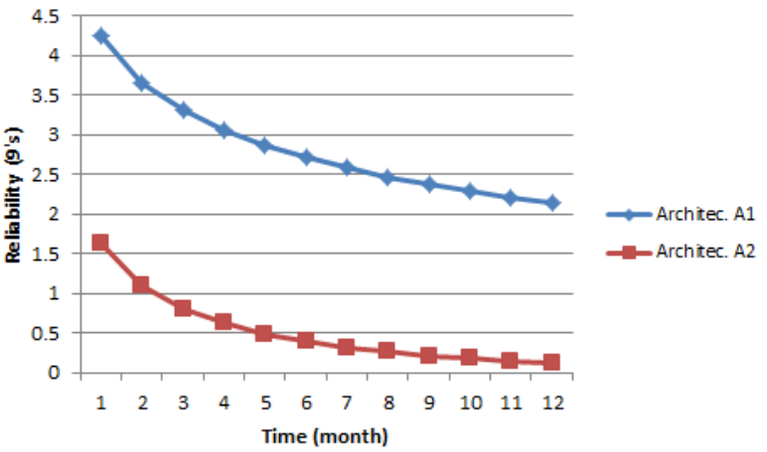


**Figure 23.** Reliability Comparison of Architectures A1 and A2.

## 6. Conclusion

This work considers the advantages of both Stochastic Petri Nets (SPN) and Reliability Block Diagrams (RBD) formalisms to analyze data center infrastructures. Such approach is supported by an integrated environment, ASTRO, which allows data center designers to estimate the dependability metrics before implementing the architectures. The methodology proposes that the system should be evaluated piecewisely to allow the composition of simpler models representing a data center infrastructure appropriately. Moreover, experiments demonstrate the feasibility of the environment, in which different architectures for a data center power infrastructures have been adopted.

## Acknowledgments

The authors would like to thank CNPQ for financing the project (290018/2011-0) and supporting the development of this work.

## Author details

Gustavo Callou, Paulo Maciel, Julian Araújo, João Ferreira and Rafael Souza
*Informatics Center, Federal University of Pernambuco - Recife, Brazil*

Dietmar Tutsch
*Automation/Computer Science, University of Wuppertal, Wuppertal, Germany*

## 7. References

[1] Avizienis, A., Laprie, J. & Randell, B. [2001]. Fundamental Concepts of Dependability, *Technical Report Series-University of Newcastle upon Tyne Computing Science* .

[2] Banerjee, P., Bash, C., Friedrich, R., Goldsack, P., Huberman, B. A., Manley, J., Patel, C., Ranganathan, P. & Veitch., A. [2011]. Everything as a service: Powering the new information economy, *IEEE Computer* pp. 36–43.

[3] Callou, G., Maciel, P., Magnani, F., Figueiredo, J., Sousa, E., Tavares, E., Silva, B., Neves, F. & Araujo, C. [2011]. Estimating sustainability impact, total cost of ownership and dependability metrics on data center infrastructures, *Sustainable Systems and Technology (ISSST), 2011 IEEE International Symposium on*, pp. 1 –6.

[4] Callou, G., Maciel, P., Tavares, E., Sousa, E., Silva, B., Figueiredo, J., Araujo, C., Magnani, F. & Neves, F. [2011]. Sustainability and dependability evaluation on data center architectures, *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, pp. 398 –403.

[5] Callou, G., Sousa, E., Maciel, P., Tavares, E., Silva, B., Figueirêdo, J., Araujo, C., Magnani, F. & Neves, F. [2011]. A formal approach to the quantification of sustainability and dependability metrics on data center infrastructures, *Proceedings of the 2011 Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, TMS-DEVS '11, Society for Computer Simulation International, San Diego, CA, USA, pp. 274–281.
URL: *http://dl.acm.org/citation.cfm?id=2048476.2048512*

[6] Desrochers, A. & Al-Jaar, R. [1995]. *Applications of Petri Nets in Manufacturing Systems: Modeling, Control, and Performance Analysis*, IEEE Press.

[7] Dutuit, Y., Châtelet, E., Signoret, J. & Thomas, P. [1997]. Dependability modelling and evaluation by using stochastic Petri nets: application to two test cases, *Reliability Engineering & System Safety* 55(2): 117–124.

[8] Ebeling, C. [1997]. *An Introduction to Reliability and Maintainability Engineering*, Waveland Press.

[9] German, R. [2000]. *Performance Analysis of Communication Systems with Non-Markovian Stochastic Petri Nets*, John Wiley & Sons, Inc., New York, NY, USA.

[10] Kuo, W. & Zuo, M. J. [2003]. *Optimal Reliability Modeling - Principles and Applications*, Wiley.

[11] Marsan, M. A., Balbo, G., Conte, G., Donatelli, S. & Franceschinis, G. [1995]. *Modelling with Generalized Stochastic Petri Nets.*, John Wiley and Sons.

[12] Marwah, M., Maciel, P., Shah, A., Sharma, R., Christian, T., Almeida, V., Araújo, C., Souza, E., Callou, G., Silva, B., Galdino, S. & Pires, J. [2010]. Quantifying the sustainability impact of data center availability, *SIGMETRICS Perform. Eval. Rev.* 37: 64–68.
URL: *http://doi.acm.org/10.1145/1773394.1773405*

[13] Meyer, J. F. & Sanders, W. H. [1993]. Specification and construction of performability models., *Proceedings of the Second International Workshop on Performability Modeling of Computer and Communication Systems, Mont Saint-Michel, France*.

[14] Murata, T. [1989]. Petri nets: Properties, analysis and applications, *Proceedings of the IEEE* 77(4): 541 –580.

[15] Patterson, D. [2002]. A simple way to estimate the cost of downtime, *Proceedings of the 16th USENIX conference on System administration*, LISA '02, USENIX Association, Berkeley, CA, USA, pp. 185–188. URL: *http://dl.acm.org/citation.cfm?id=1050517.1050538*

[16] Petri, C. A. [1962]. *Kommunikationmit Automaten*, PhD Dissertation,Darmstad University, Germany.

[17] Reisig, W. [1985]. *Petri nets: an introduction*, Springer-Verlag New York, Inc., New York, NY, USA.

[18] Robidoux, R., Xu, H., Member, S., Xing, L., Member, S. & Zhou, M. [2010]. Automated modeling of dynamic reliability block diagrams using colored petri nets, *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans* 40(2): 337–351.

[19] *Service Level Agreement for Data Center Services* [2012]. http://www.earthlinkbusiness.com/_static/_files/_pdfs/legal/DataCenterServiceSLA.pdf.

[20] Silva, B., Maciel, P., Tavares, E., Araujo, C., Callou, G., Sousa, E., Rosa, N., Marwah, M., Sharma, R., Shah, A., Christian, T. & Pires, J. [2010]. Astro: A tool for dependability evaluation of data center infrastructures, *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pp. 783 –790.

[21] std., I. [1997]. *Gold Book 473 Design of Reliable Industrial and Commercial Power Systems*, IEEE.

[22] Trivedi, K. [2002]. *Probability and Statistics with Reliability, Queueing, and Computer Science Applications*, 2 edn, Wiley Interscience Publication.

[23] Trivedi, K. & et al [1994]. Reliability analysis techniques explored through a communication network example, *International Workshop on Computer-Aided Design, Test, and Evaluation for Dependability*.

[24] Vilkomir, S. A., Parnas, D. L., Mendiratta, V. B. & Murphy, E. [2006]. Segregated failures model for availability evaluation of fault-tolerant system, *ACSC '06: Proceedings of the 29th Australasian Computer Science Conference*, Australian Computer Society, Inc., Darlinghurst, Australia, Australia, pp. 55–61.

[25] Wei, B., Lin, C. & Kong, X. [2011]. Dependability modeling and analysis for the virtual data center of cloud computing, *High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on*, pp. 784 –789.

[26] Wiboonrat, M. [2008a]. An empirical study on data center system failure diagnosis, *Internet Monitoring and Protection, 2008. ICIMP '08. The Third International Conference on*, pp. 103 –108.

[27] Wiboonrat, M. [2008b]. Risk anatomy of data center power distribution systems, *ICSET'08*.

[28] Xu, H., Xing, L. & Robidoux, R. [2008]. Drbd: Dynamic reliability block diagrams for system reliabilitymodeling, *International Journal of Computers and Applications* .

[29] Zhou, L. & Grover, W. [2005]. A theory for setting the "safety margin" on availability guarantees in an sla, *Design of Reliable Communication*