Chapter 2

Modeling Strategies to Improve the Dependability of Cloud Infrastructures

Erica Teixeira Gomes de Sousa and
Fernando Antonio Aires Lins

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.71498

Abstract

Cloud computing presents some challenges that need to be overcome, such as planning infrastructures that maintain availability when failure events and repair activities occur. Cloud infrastructure planning that addresses the dependability aspects is an essential activity because it ensures business continuity and client satisfaction. Redundancy mechanisms cold standby, warm standby and hot standby can be allocated to components of the cloud infrastructure to maintain the availability levels agreed in service level agreement (SLAs). Mathematical formalisms based on state space such as stochastic Petri nets and based on combinatorial as reliability block diagrams can be adopted to evaluate the dependability of cloud infrastructures considering the allocation of different redundancy mechanisms to its components. This chapter shows the adoption of the mathematical formalisms stochastic Petri nets and reliability block diagrams to dependability evaluation of cloud infrastructures with different redundancy mechanisms.

Keywords: dependability evaluation, state space models, non-state space models, redundancy mechanisms, maintenance policies

1. Introduction

Ensuring the availability levels required by the different services hosted in the private cloud is a great challenge. The occurrence of defects in these services can cause the degradation of their response times and the interruption of service of a request due to unavailability of the required resource. The interruption of these services can be caused by the occurrence of failure events in the hardware, software, power system, cooling system and private cloud network. When the occurrence of defects is constant, users give less preference to hiring service providers due to reduced availability, reliability and performance of these services [1].

The dependability assessment can minimize the occurrence of faults and failure events [2] in the private cloud and promote the levels of availability and reliability defined in the SLAs, avoiding the payment of contractual fines. One option to ensure the availability of services offered in the private cloud is to assign redundant equipment to its components. Redundant devices allow service reestablishment, minimizing the effects of failure events. The major problem with this assignment is the estimation of the number of redundant equipment and the choice of the type of redundancy that must be considered to guarantee the quality of the service offered. The estimation of the type and number of redundant equipment should also consider the cost of the quantitative of each type of redundancy mechanism attributed to the components of the cloud computing [3, 4].

2. Basic concepts

The dependability evaluation denotes the ability of a system to deliver a reliably service. Dependability measures are reliability, availability, maintainability, performability, safety, testability, confidentiality, and integrity [2].

Dependability evaluation is related to the study of the effect of errors, defects and failures in the system, since these have a negative impact on the dependability attributes. A fault is defined as the failure of a component, subsystem or system that interacts with the system in question [5]. An error is defined as a state that can lead to a failure. A defect represents the deviation from the correct operation of a system. A summary of the main measures of dependability is shown below.

The reliability of a system is the probability (P) that this system performs its function satisfactorily, without the occurrence of defects, for a certain period of time (T). Reliability is represented by Eq. (1), where T is a random variable that represents the time for occurrence of defects in the system [3, 4].

$$R(t) = P\{T > t\}, t \geq 0 \tag{1}$$

The probability of the occurrence of defects up to a time t, is represented by Eq. (2), where T is a random variable that represents the time for system failures [3, 4].

$$F(t) = 1 - R(t) = P\{T \leq t\} \tag{2}$$

Eq. (3) represents the reliability, considering the density function F(t) of the time for occurrence of failures (T) in the system [3, 4, 6].

$$R(t) = P\{T > t\} = \int_t^{\infty} F(t)dt \tag{3}$$

The Mean Time to Failure (MTTF) is the average time for defects to occur in the system. When this average time follows the exponential distribution with parameter λ, the MTTF is represented by Eq. (4) [3, 4, 6].

http://dx.doi.org/10.5772/intechopen.71498

$$\text{MTTF} = \int_0^\infty \text{R(t)dt} = \int_0^\infty \text{e}^{(-\lambda)\text{t}} = \frac{1}{\lambda} \tag{4}$$

The failures can be classified in relation to the time, according to the mechanism that originated them. The behavior of the failure rate can be represented graphically through the bathtub curve, which presents three distinct phases: infant mortality (1), useful life (2) and aging (3). **Figure 1** shows the variation of the failure rate of hardware components as a function of time [7].

During the infant mortality phase (1), a reduction in the failure rate occurs. Failures during this period are due to equipment manufacturing defects. In order to shorten this period, manufacturers submit the equipment to a process called burn-in, where they are exposed to high operating temperatures. In the useful stage (2), the failures occur randomly. Equipment reliability values provided by manufacturers apply to this period. The service life of the equipment is not normally a constant. It depends on the level of stress in which the equipment is subjected during that period. During the aging phase (3), an increase in the failure rate occurs.

In high availability environments, one must be sure that the infant mortality phase has passed. In some cases, it is necessary to leave the equipment running in a test environment during this time. At the same time, care must be taken to have the equipment replaced before entering the aging phase.

The availability of a system is the probability that this system is operational for a certain period of time, or has been restored after a defect has occurred. Uptime is the period of time in which the system is operational, downtime is the period of time when the system is not operational due to a defect or repair activity occurring, and uptime + downtime is the time period of observation of the system. Eq. (5) represents the availability of a system [3, 4, 6].

$$\text{A} = \frac{\text{uptime}}{\text{uptime} + \text{downtime}} \tag{5}$$

Computational systems and applications require different levels of availability and therefore can be classified according to these levels. U.S. Federal Aviation Administration's National Airspace System's Reliability Handbook classifies computer systems and applications according to their criticality levels [1]. These computational systems and applications can be

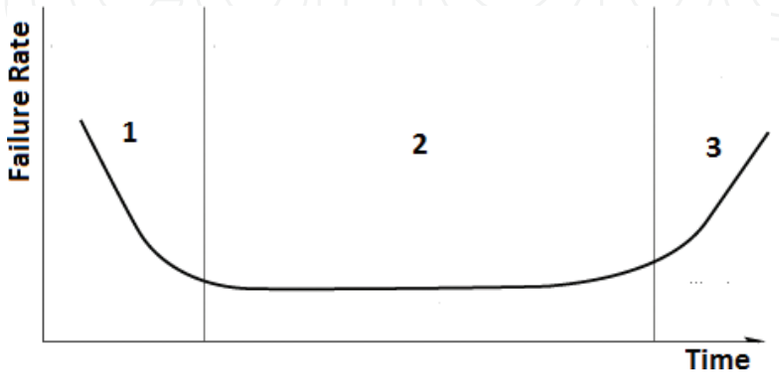

Figure 1. Bathtub curve.

considered critical critics when the required availability is 99.99999%, critical when the required availability is 99.999%, essential when the required availability is 99.9% and routine when the required availability is 99% [1].

The maintainability is the probability that a system can be repaired in a given period of time (TR). The maintainability is described by Eq. (6), where TR denotes the repair time. This equation represents maintainability, since the repair time TR has a density function G(t) [3, 4, 6].

$$V(t) = P\{TR \leq tr\} = \int_0^{tr} G(t)dt \tag{6}$$

The Mean Time to Repair (MTTR) is the average time to repair the system. When the time distribution function of repair is represented by an exponential distribution with parameter μ, the MTTR is represented by Eq. (7) [3, 4, 6].

$$MTTR = \int_0^{\infty} 1 - G(tr)dt = \int_0^{\infty} 1 - e^{(\mu)tr} = \frac{1}{\mu} \tag{7}$$

Mean Time Between Failures (MTBF) is the mean time between system defects, represented by Eq. (8) [3, 4, 6].

$$MTBF = MTTR + MTTF \tag{8}$$

Performability describes the degradation of system performance caused by the occurrence of defects [3, 6].

3. Redundancy mechanisms

The redundancy mechanisms provide greater availability and reliability to the system during the occurrence of failure events due to the maintenance of components operating in parallel, that is, a redundant system has a secondary component that will be available when the primary component fails. Thus, redundancy mechanisms are designed to avoid single points of failure and therefore provide high availability and disaster recovery if necessary [1, 8].

The redundancy mechanisms can be classified as active-active and active-standby. Active-active redundancy mechanisms are employed when the primary and secondary components share the workload of the system. When any of these components fails, the other component will be responsible for servicing the system users' requests. These redundancy mechanisms can be classified as N + K, where K secondary components identical to N primary components are required for system workload sharing. In the N + 1, configuration, a secondary component identical to the primary N components is required for sharing the system workload. In the N + 2, configuration, two secondary components identical to the N primary components are required for system workload sharing [1].

The active-standby redundancy mechanisms are employed when the primary components meet the requests of the system users and the secondary components are on hold. When the primary components fail, the secondary components will be responsible for servicing the system users' requests. The active-standby redundancy mechanisms can be classified as hot standby, cold standby and warm standby [1].

In the hot standby redundancy mechanism, redundant modules that are in standby function in synchronization with the operating module, without their computation being considered in the system, and in case the occurrence of a failure event is detected, it is ready to make operational immediately [1, 4].

In the cold standby redundancy mechanism, the redundant modules are turned off and only when a failure event occurs will they be activated after a time interval. In the cold standby redundancy mechanism, inactive modules that are de-energized, by hypothesis, do not fail, whereas the active module has a constant failure rate λ.

In the warm standby redundancy mechanism, the redundant modules that are in standby function in sync with the operating module, without their computation being considered in the system. If a fault event is detected, the redundant module is ready to become operational after a time interval. Systems with standby sparing of cold standby sparing and warm standby sparing need more time for recovery compared to hot standby sparing, but systems with cold standby sparing and warm standby sparing have the advantage of lower power consumption and no wear standby systems [1, 4].

4. Modeling techniques

The models adopted for dependability evaluation can be classified as combinatorial and state space. The combinatorial models capture the conditions that cause failures in the system or allow its operation when considering the structural relationships of its components. The best known combinatorial models are Reliability Block Diagram (RBD) and Fault Tree (FT) [9, 10]. State space-based models represent the behavior of the system (occurrence of failures and repair activities) through its states and the occurrence of events. These models allow the representation of dependency relations between the components of the systems. The most widely used state space-based models are Markov Chains (MC) and Stochastic Petri Net (SPN) [9–12].

The SPN models provide great flexibility in the representation of aspects of dependability. However, these models suffer from problems related to the size of the state space for computational systems with large number of components [11, 12]. RBD models are simple, easy to understand, and their solution methods have been extensively studied. These models can represent the components of cloud computing that do not have a dependency relation to allow an efficient representation, avoiding growth problems too much of the space of states [9, 10].

5. Reliability block diagram

Reliability block diagram (RBD) is one of the most used techniques for reliability analysis of systems [5].

The RBD allows the calculation of availability and reliability by means of closed formulas, since it is a combinational model. These closed formulas make the calculation of the result faster than the simulation, for example [6].

In a reliability block diagram, components are represented with blocks combined with other blocks (i.e., components) in series, parallel or combinations of those structures. A diagram that has components connected in series requires each component to be running for the system to be operational. A diagram that has components connected in parallel requires that only one component is working for the system to be operational [13]. Thus, the system is described as a set of interconnected functional blocks to represent the effect of availability and reliability of each block on the availability and reliability of the system [14].

The availability and reliability of two blocks connected in series is obtained through Eq. (9) [6].

$$P_s = \prod_{i=1}^{n} P_i(t) \tag{9}$$

where:

Pi(t) describes the reliability Ri(t), the instantaneous availability Ai(t) e a and the steady state availability Ai of the block Bi.

The availability and reliability of two blocks connected in parallel is obtained through Eq. (10) [6].

$$P_p = 1 - \prod_{i=1}^{n} (1 - P_i(t)) \tag{10}$$

where Pi(t) describes the reliability Ri(t), the instantaneous availability Ai(t) e a and the steady state availability Ai of the block Bi.

Figure 2 shows the connection of the blocks in series and **Figure 3** shows the connection of the blocks in parallel.

The reliability block diagram is mainly used in modular systems consisting of many independent modules, where each can be easily represented by a block.

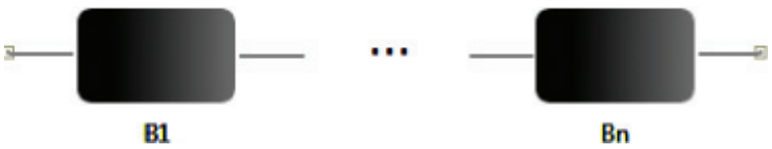

Figure 2. Reliability block diagram in series.

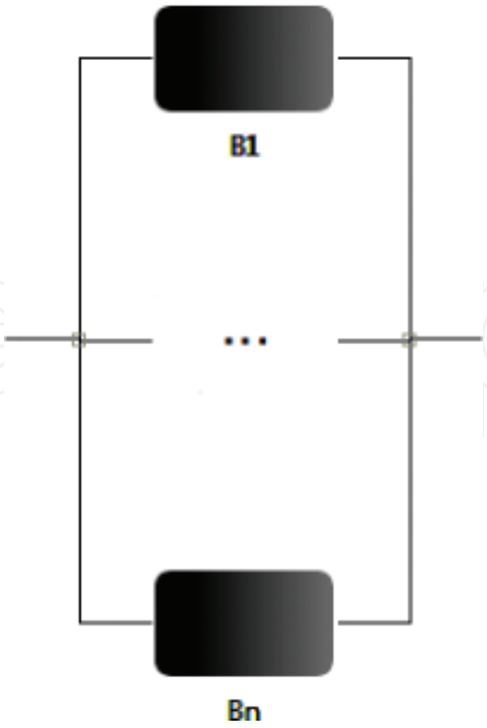

Figure 3. Reliability block diagram in parallel.

6. Petri nets

The concept of Petri nets was introduced by Carl Adam Petri in 1962 with the presentation of his doctoral thesis "Kommunikation mit Automaten" (Communication with Automata) [15] at the Faculty of Mathematics and Physics of Darmstadt University in Germany. Petri nets are graphical and mathematical tools used for formal description of systems characterized by properties of concurrency, parallelism, synchronization, distribution, asynchronism, and non-determinism [15].

The applicability of Petri nets as a tool for systems study is important because it allows for mathematical representation, analysis of models and also for providing useful information about the structure and dynamic behavior of the modeled systems. The applications of Petri nets can occur in many areas (systems of manufacture, development and testing of software, administrative systems, among others) [16].

The Petri nets presents some characteristics that are: the dynamic representation of the modeling system with the desired level of detail; The graphical and formal description that allows to obtain information on the behavior of the modeled system through its behavioral and structural properties; The representation of synchronism, asynchronism, competition, resource sharing, among other behaviors; And the wide applicability and documentation.

Petri nets are formed by places (1), transitions (2), arcs (3) and marking (4). The places correspond to state variables and the transitions, actions or events performed by the system. The performance of an action is associated with some preconditions, that is, there is a relation

between the places and the transitions that allows or not the accomplishment of a certain action. After performing a certain action, some places will have their information changed, that is, the action will create a post condition. The arcs represent the flow of the marking through the Petri net, and the tokens represent the state in which the system is at a given moment. Graphically, places are represented by ellipses or circles, transitions, by rectangles, arcs, by arrows and marking, by means of dots (**Figure 4**) [16].

The two elements, place and transition, are interconnected by directed arcs as shown in **Figure 5**. The arcs that interconnect places to the transitions (Place → Transition) correspond to the relationship between the true conditions (precondition), which enable the execution Of the shares. The arcs that interconnect transitions to places (Transition → Place) represent the relationship between actions and conditions that become true with the execution of actions (post condition) [16].

The formal mathematical representation of a model in Petri net (Petri net—PN) is the quintuple PN = P, T, F, W, M0 [15], where:

6.1. Properties of Petri nets

The study of the properties of Petri nets allows the analysis of the modeling system. Property types can be divided into two categories: the initial marking-dependent properties, named behavioral properties, and the non-marking properties, named structural properties [15, 16].

6.1.1. Behavioral properties

The behavioral properties are those that depend only on the initial marking of the Petri net. The properties covered are reachability, limitation, safeness, liveness and coverage.

Reachability indicates the possibility that a given marking can be reached by firing a finite number of transitions from an initial marking. Given a Petri net marked RM = (R,M0), the triggering of a transition t0 alters the marking of the Petri net. An M′ label is accessible from M0 if there is a sequence of transitions which, triggered, lead to the M′ label. That is, if the marking M0 enables the transition t0, by triggering this transition, the marking M1 is reached. The marking M1 enables t1 which, upon being triggered, reaches the marking M2 and so on until the marking M′ is obtained.

Figure 4. Elements of Petri net.

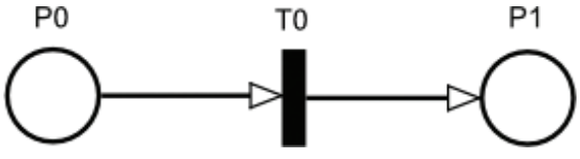

Figure 5. Example of Petri net.

Let M a place pi ∈ P, of a Petri net marked RM = (R, M0), this place is k-bounded (k ∈ IN) or simply limited if for every accessible marking M ∈ CA (R, M0), M (pi) ≤ k.

The limited k is the maximum number of marking that a place can accumulate. A Petri net labeled RM = (R, M0) is k-bounded if the number of marking at each RM site does not exceed k at any accessible RM marking (max (M (p)) = k, ∀ p ∈ P).

Safeness is a particularization of limited property. The concept of limited defines that a pi place is k-bounded if the number of marking that this place can accumulate is limited to the number k. A place that is 1-limited can simply be called insurance.

Liveness is defined according to the triggering possibilities of the transitions. A Petri net is considered live if, regardless of the marking that are reachable from M0, it is always possible to trigger any transition of the Petri net through a sequence of transitions L(M0). The absence of deadlock in systems is strongly linked to the concept of vivacity, since deadlock in a Petri net is the impossibility of triggering any transition of the Petri net. The fact that a system is deadlock free does not mean that it is live, however a live system implies a deadlock free system.

The concept of coverage is associated with the concept of reachability and live. An Mi marking is covered if there is a marking Mj ≠ Mi, such that Mj ≥ Mi.

6.1.2. Structural properties

The structural properties are those that depend only on the structure of the Petri net. These properties reflect independent marking characteristics. The properties analyzed in this work are structural limitation and consistency.

A Petri net R = (P, T, F, W, M0) is classified as structurally limited if it is limited to any initial marking.

The Petri net is considered to be consistent if, by triggering a sequence of enabled transitions from an M0 marking, it returns to M0, however all transitions of the Petri net are fired at least once.

Let RM = (R, M0) be a marked Petri net and a sequence s of transitions, RM is consistent if M0 [s > M0] and every transition Ti, firing at least once in s.

6.2. Stochastic Petri net

Petri Net (SNP) [11] is one of the Petri net extensions (PN) [15] used for performance and dependability modeling. A stochastic Petri net adds time to Petri net formalism, with the difference that the times associated with the timed transitions are exponentially distributed, while the time associated with the immediate transitions is zero. The timed transitions model activities through the associated times, so that the timing transition period corresponds to the activity execution period, and the timed transition trigger corresponds to the end of the activity. Different levels of priority can be assigned to transitions. The trigger priority of the immediate transitions is higher than the timed transitions. Priorities can solve situations of confusion [12]. The firing probabilities associated with immediate transitions can resolve conflict situations [4, 5].

Timed transitions can be characterized by different memory policies such as Resampling, Enabling memory and Age memory [5]. The timed transitions can also be characterized by different firing semantics named single server, multiple server and infinite server [5].

6.3. Phase approximation technique

SPN models consider only immediate transitions and timed transitions with exponentially distributed trigger times. These transitions model actions, activities, and events. A variety of activities can be modeled through the use of constructor throughput subnets and s-transitions. These constructs are used to represent expolinomial distributions, such as the Erlang, hypoexponential and hyperexponential distributions [9].

The phase approximation technique can be applied to model non-exponential actions, activities, and events through moment matching. The presented method calculates the first moment around the origin (average) and the second central moment (variance) and estimates the respective moments of the s-transition [9].

Performance and dependability data measured or obtained from a system (empirical distribution) with mean μ and standard deviation σ may have their approximate stochastic behavior through the phase approximation technique. The inverse of the variation coefficient of the data measured or obtained from a system Eq. (11) allows the selection of the expolinomial distribution that best adapts to the empirical distribution. This empirical distribution can be continuous or discrete. Among the continuous distributions, there are: Normal, Lognormal, Weibull, Gamma, Continuous Uniform, Pareto, Beta and Triangular and among the discrete distributions there are: Geometric, Poisson and Discrete Uniform [8].

$$\frac{1}{CV} = \frac{\mu_D}{\sigma_D} \tag{11}$$

The Petri net described in **Figure 6** represents a timed activity with generic probability distribution.

Depending on the inverse of the variation coefficient of the measured data (Eq. (11)), the respective activity has one of these distributions attributed: Erlang, Hypoexponential or Hyperexponential. When the inverse of the variation coefficient is an integer and different from one, the data must be characterized by the Erlang distribution. When the inverse of the variation coefficient is a number greater than one (but not an integer), the data are represented by the hypoexponential distribution. When the inverse of the variation coefficient is a number smaller than one, the data must be characterized by a hyperexponential distribution.

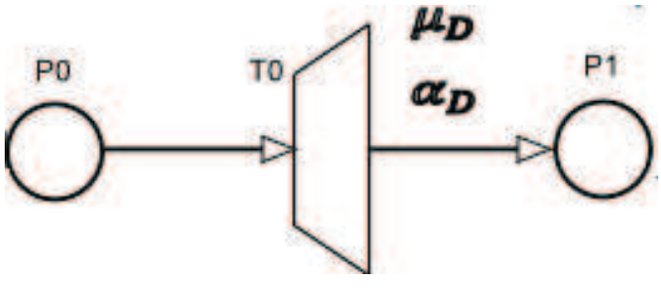

Figure 6. Empirical distribution.

http://dx.doi.org/10.5772/intechopen.71498

7. Modeling strategy

The dependability metrics can be calculated using state space-based models (e.g., SPN) and combinatorial models (e.g., RBD). The RBDs have an advantage over the provision of results, as present faster calculations through their formulas than the simulations and the numerical analyzes of the SPNs. However, SPNs have a greater power of representation [3, 17].

State space-based models can describe dependencies that allow the representation of complex redundancy mechanisms. However, these models can generate a very large or even infinite number of states when they represent highly complex systems [3, 12, 17].

The combination of state space-based models and combinatorial models allows for the reduction of complexity in the representation of systems. RBD models can represent the components of the cloud computing [6]. These RBD models are used to estimate the availability and downtime of the cloud computing when there is little dependency relation between the components of this environment and the redundancy mechanisms adopted. If there is a need to represent a greater dependency between the components of the cloud computing and the redundancy mechanisms used, SPN models are used to represent the computational cloud systems [11].

Figure 7 shows a basic SPN model that allows a representation of the cloud computing. In this SPN model, the ON and OFF places represent the working or faulted computational cloud. The attributes of the transitions of this SPN model are presented in **Table 1**.

Figure 8 shows a basic RBD model that allows a representation of the cloud computing. The parameters of the RBD model are presented in the **Table 2**.

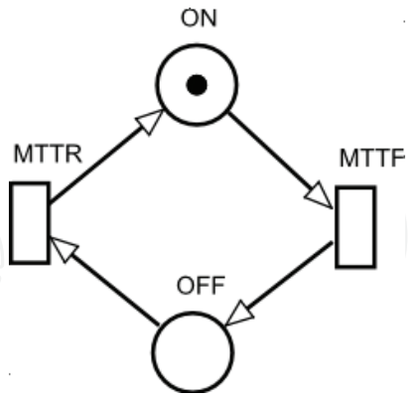

Figure 7. Basic SPN model.

Transition	Type	Time	Weight	Concurrence
MTTF	exp	XMTTF	–	SS
MTTR	exp	XMTTR	–	SS

Table 1. Attributes of the SPN model transitions.

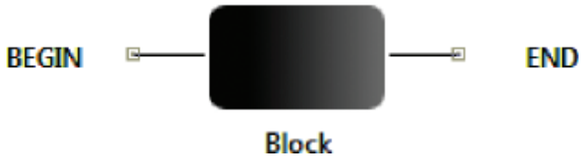

Figure 8. Basic RBD model.

Parameters	Description
MTTFBlock	Mean Time to Failure
MTTRBlock	Mean Time to Repair

Table 2. Parameters of the RBD model.

7.1. Cloud computing model

Cloud computing consists of the Cloud Controller (Controller), Node Controller (Node) and Network equipment (Network). **Figure 9** shows the RBD model of the cloud computing. The parameters of the RBD model of cloud computing are presented in **Table 3**. **Figure 10** shows the SPN model of the cloud computing. The attributes of the SPN Model Transitions of cloud computing are presented in **Table 4**. In RBD and SPN models, the cloud controller and node controller are configured on different physical machines. The node controller enables the instantiation of virtual machines. The physical machines where the components of cloud computing are configured are connected through a switch and a router. All components of cloud computing must be operational for the cloud computing to be operational. These components can be described as Controller, Node, and Network. In this way, the operating mode of cloud computing is OM = (Controller ∧ Node ∧ Network).

Cloud computing consists of the Cloud Controller (Controller), Node Controller (Node) and Network equipment (Network). The Cloud Controller (Controller) has a hot standby redundancy, but the other components (Node and Network) can also be assigned this redundancy. The main cloud controller (ControllerMain) and the redundant cloud controller (ControllerStandby) in hot standby are operational [3, 4]. The operating mode of cloud computing with redundant cloud

Figure 9. RBD model of the cloud computing.

Parameters	Description
MTTFController, MTTFNode, MTTFNetwork	Mean Time to Failure of the controller, node and network
MTTRController, MTTRNode, MTTRNetwork	Mean Time to Repair of the controller, node and network

Table 3. Parameters of the RBD model of the cloud computing.

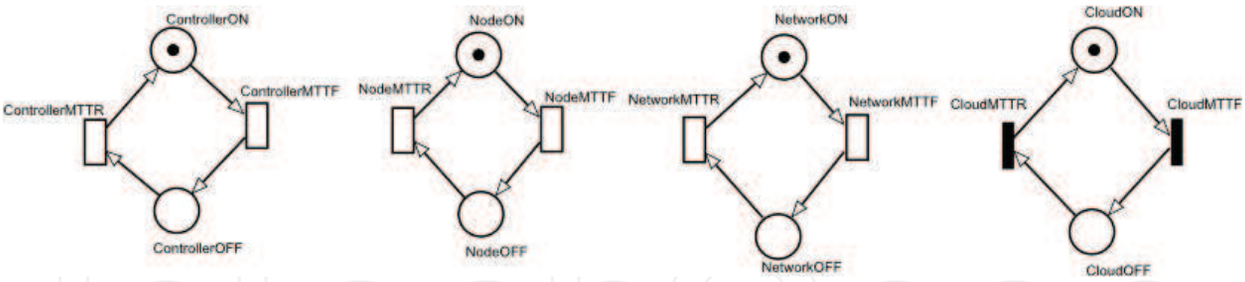

Figure 10. SPN model of the cloud computing.

Transition	Type	Time	Weight	Concurrence	Enable function
ControllerMTTF, NodeMTTF, NetworkMTTF	exp	XMTTF	–	SS	–
ControllerMTTR, NodeMTTR, NetworMTTR	exp	XMTTR	–	SS	–
CloudMTTF	imme	–	1	–	((#ControllerON = 0)OR(#NodeON = 0)OR (#NetworkON = 0))
CloudMTTR	imme	–	1	–	NOT((#ControllerON = 0)OR(#NodeON = 0) OR(#NetworkON = 0))

Table 4. Attributes of the SPN model transitions of the cloud computing.

controller in hot standby is OM = ((ControllerMain ∨ ControllerStandby) Λ Node Λ Network)). **Figure 11** shows the RBD model adopted to estimate the availability of cloud computing with redundant cloud controller in hot standby.

Cloud computing consists of the Cloud Controller (Controller), Node Controller (Node) and Network equipment (Network). The Cloud Controller (Controller) has a cold standby redundancy, but the other components (Node and Network) can also be assigned this redundancy. The main cloud controller (ControllerMain) is operational and the redundant cloud controller (ControllerStandby) is non-active. The redundant cloud controller is not operational waiting to

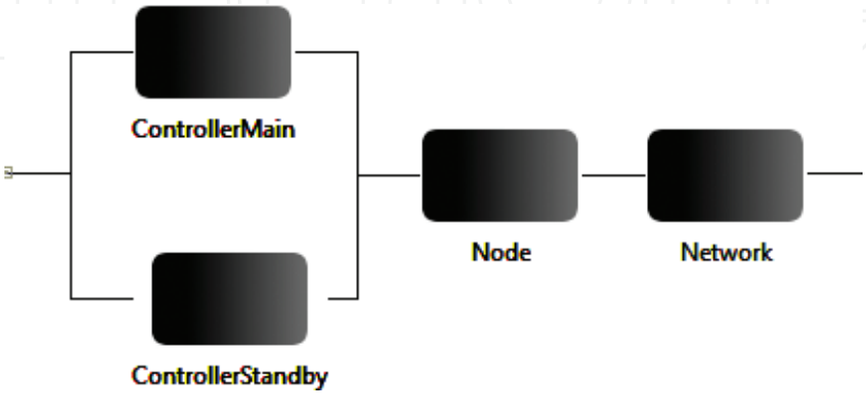

Figure 11. RBD model of the cloud computing with redundant cloud controller in hot standby.

be activated when the main cloud controller fails. Thus, when the main cloud controller, the activation of the redundant cloud controller occurs in a certain period of time. This period is named Mean Time to Active (MTA) [3, 4]. The operating mode of cloud computing with redundant cloud controller in cold standby is OM = ((ControllerMain ∨ ControllerStandby) Λ Node Λ Network)). **Figure 12** shows the SPN model adopted to estimate the availability of cloud computing with redundant cloud controller in cold standby.

Cloud computing consists of the Cloud Controller (Controller), Node Controller (Node) and Network equipment (Network). The Cloud Controller (Controller) has a warm standby redundancy, but the other components (Node and Network) can also be assigned this redundancy. The main cloud controller (ControllerMain) is based on a non-active redundant cloud controller (ControllerStandby) that waits to be activated when the main cloud controller fails. The difference with respect to cold standby redundancy is that the main cloud controller and the redundant cloud controller have an λ failure rate when they are in operation, but the redundant cloud controller has a failure rate φ when it is de-energized, considering that $0 \leq \lambda \leq \varphi$ [3, 4]. The redundant cloud controller (ControllerStandby) starts in idle mode. When the main cloud controller (ControllerMain) fails, the timed SpareActive transition triggers. This fire represents the start of the redundant cloud controller operation. The time associated with the SpareActive timed transition represents the Mean Time to Active (MTA). The SpareNActive immediate transition represents the return of the main module to the operational mode. The operating mode of cloud computing with redundant cloud controller in warm standby is OM = ((ControllerMain ∨ ControllerStandby) Λ Node Λ Network)). **Figure 13** shows the SPN model adopted to estimate the availability of cloud computing with redundant cloud controller in warm standby.

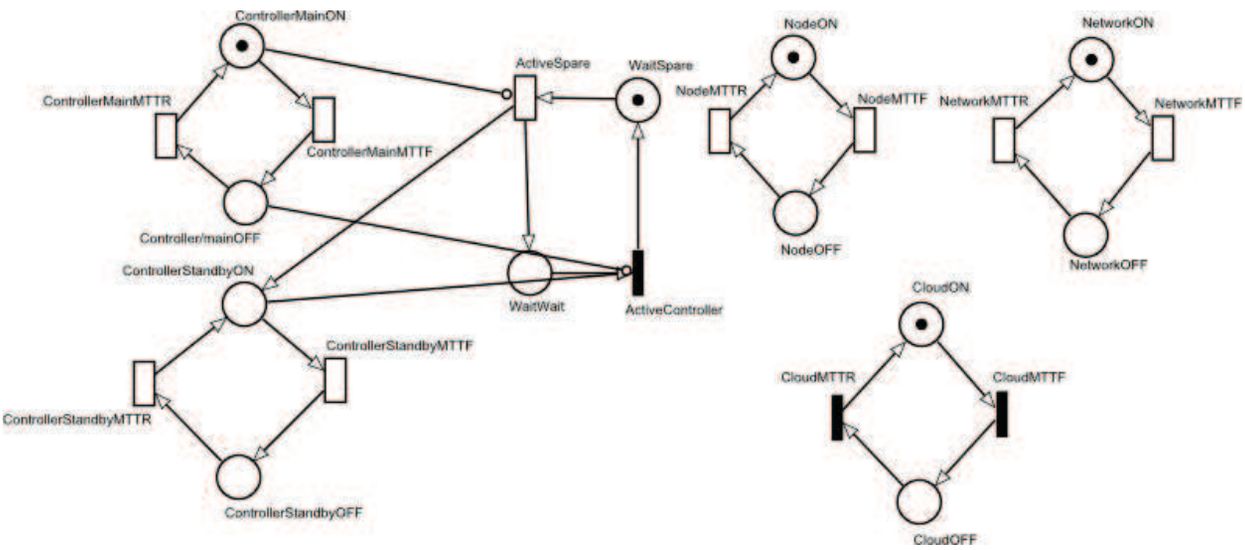

Figure 12. SPN model of the cloud computing with redundant cloud controller in cold standby.

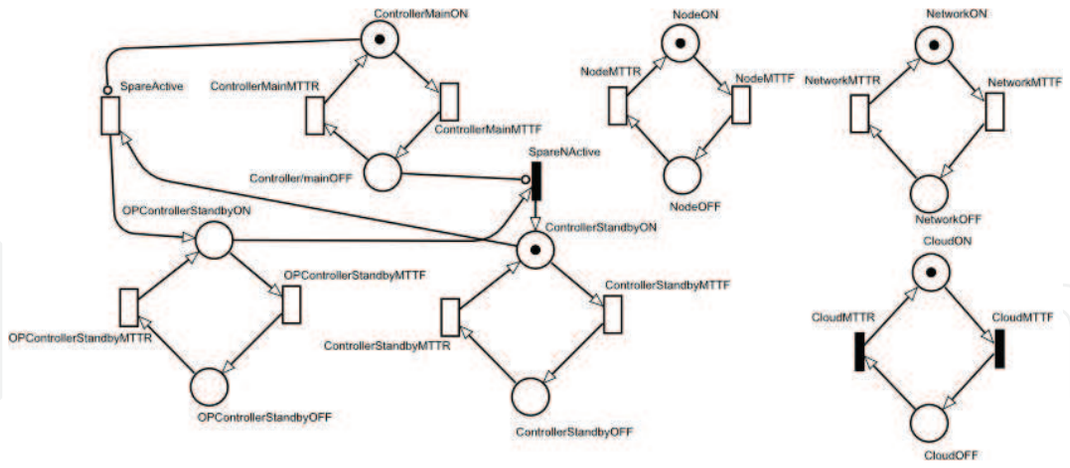

Figure 13. SPN model of the cloud computing with redundant cloud controller in warm standby.

8. Conclusions

This chapter presents concepts on dependability, redundancy mechanisms, stochastic Petri nets and reliability block diagram. In addition, this chapter also shows how the mathematical formalisms stochastic Petri nets and reliability block diagrams can be adopted for modeling cloud infrastructures with cold standby, warm standby and hot standby redundancy mechanisms. Reliability block diagrams is adopted to model cloud infrastructures with the redundancy mechanism cold standby and stochastic Petri nets is used to model cloud infrastructures with the redundancy mechanisms warm standby and hot standby.

Author details

Erica Teixeira Gomes de Sousa* and Fernando Antonio Aires Lins

*Address all correspondence to: erica.sousa@ufrpe.br

Department of Statistics and Informatics, Federal Rural University of Pernambuco, Brazil

References

[1] Bauer E, Adams R. Reliability and Availability of Cloud Computing. Wiley Online Library; 2012

[2] Laprie JCC, Avizienis A, Kopetz H. Dependability: Basic Concepts and Terminology. Secaucus, NJ, USA: Springer-Verlag New York, Inc; 1992

[3] Kuo W, Zuo MJ. Optimal Reliability Modeling: Principles and Applications. Wiley; 2002

[4] Rupe JW. Reliability of computer systems and networks fault tolerance, analysis, and design. IIE Transactions. 2003;**35**(6):586-587

[5] Maciel P, Trivedi K, Matias R, Kim D. Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions. IGI Global; 2011

[6] Xie M, Dai YS, Poh KL. Computing System Reliability: Models and Analysis. US: Springer; 2004

[7] Ebeling CE. An Introduction to Reliability and Maintainability Engineering. Waveland Pr Inc; 2009

[8] Schmidt K. High Availability and Disaster Recovery: Concepts, Design, Implementation. Vol. 22. Berlin Heidelberg: Springer-Verlag; 2006

[9] Sahner RA, Trivedi K, Puliafito A. Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the SHARPE Software Package. New York, US: Springer; 1996

[10] Trivedi KS. Probability & Statistics with Reliability, Queuing and Computer Science Applications. 2nd ed. Wiley; 2001

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

[12] Marsan MA, Balbo G, Conte G, Donatelli S, Franceschinis G. Modelling with Generalized Stochastic Petri Nets, ACM SIGMETRICS Performance Evaluation Review. Vol. 26. New York, NY, USA; 1998

[13] Trivedi KS, Hunter S, Garg S, Fricks R. Reliability analysis techniques explored through a communication network example. Citeseer, International Workshop on Computer-Aided Design, Test, and Evaluation for Dependability; 1996

[14] Smith DJ. Reliability, Maintainability and Risk: Practical Methods for Engineers. Butterworth-Heinemann; 2011

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

[16] Maciel PRM, Lins RD, Cunha PRF. Introduction of the Petri Net and Applied. Campinas, SP: X Escola de Computação; 1996

[17] Balbo G. Introduction to Stochastic Petri Nets. Lectures on Formal Methods and Performance Analysis: First EEF/Euro Summer School on Trends in Computer Science. Berg en Dal, The Netherlands, July 3–7, 2000: Revised Lectures: Springer; 2000