Chapter 5

Stochastic Reward Net-based Modeling Approach for Availability Quantification of Data Center Systems

Tuan Anh Nguyen, Dugki Min and Eunmi Choi

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.74306

Abstract

Availability quantification and prediction of IT infrastructure in data centers are of paramount importance for online business enterprises. In this chapter, we present comprehensive availability models for practical case studies in order to demonstrate a state-space stochastic reward net model for typical data center systems for quantitative assessment of system availability. We present stochastic reward net models of a virtualized server system, a data center network based on DCell topology, and a conceptual data center for disaster tolerance. The systems are then evaluated against various metrics of interest, including steady state availability, downtime and downtime cost, and sensitivity analysis.

Keywords: virtualized servers system, data center system, disaster tolerant data center

1. Introduction

Data centers (DCs) have been the core-centric of modern ICT ecosystems in recent decades. Computing resources and crucial telecommunications are centralized in a data center to constantly facilitate online business and to connect people from distant parts of the world through the internet. Giant internet companies such as Facebook, Amazon, and Google have built huge state-of-the-art centers to house their own IT infrastructure. According to a study by the Ponemon Institute [1] regarding the cost of data center outages from 63 DCs located in the United States over a 12-month period, the average cost due to unplanned outages in 2016 was US$ 740,357, which steadily increased by 46% from US$ 505,502 since it was first studied in 2010. Specifically, a minute of downtime costs around US$ 7900 on average. However, online businesses actually face more severe revenue losses due to IT service downtime. In early 2016, Amazon suffered an incredible business loss of US$ 66,240/minute due to server downtime

over a period of approximately 15 minutes. The causes of system outages in DCs span from uncertain failures of IT parts/blocks to natural disasters. Therefore, a quantification of IT infrastructure availability in DCs under various scenarios in advance of system development is of paramount importance for big tech companies.

Availability assessment approaches are primarily based on measurement and modeling methods. Model-based approaches are fast and relatively inexpensive methods for system availability analysis in comparison with measurement-based methods. System modeling can be accomplished using discrete-event simulation [2, 3], analytical models, or a hybrid of both approaches. Analytical models fall into four main categories [4–7]: (i) non-state-space models (reliability graph (RelGraph), reliability block diagram (RBG), or fault tree (FT)), state-space models (Markov chains, Stochastic Petri net (SPN), stochastic reward net (SRN), etc.), hierarchical models, and fixed-point iterative models. Non-state-space modeling paradigms provide a relatively quick evaluation of basic metrics for a system (reliability, availability, MTTF) with a proper capture of overall system architecture. State-space models, on the other hand, can capture sophisticated behaviors and operations of a system. This approach can handle failure/repair dependencies and complex interactions between system components. To avoid the largeness problem (or state-space explosion problem) in state-space models, we use hierarchical modeling techniques of non-state-space and state-space models at upper and lower levels, as well as fix-point iterative models. In this chapter, we focus on studying complex system operations in DCs captured by using an SRN.

The structure of this chapter is organized into six sections. Section 2 provides preliminary concepts of availability modeling and analysis of data center systems (DCS). Subsequently, several case studies are presented. Section 3 offers an availability model of a unit system of the virtualized server (VSS) in DCs. In Section 4, we present availability modeling of a data center network (DCN) based on DCell topology. We present an SRN model for a DC in order to study disaster tolerance in Section 5. Finally, we present conclusions in Section 6.

2. Availability quantification of data center systems: basic concepts

Availability A(t) of a DCS represents the probability of its operating system taking the correct state at an instant *t*, regardless of the number of failures and repairs during the interval (*0*,*t*). *Instantaneous/point* availability *A(t)* is related to the system reliability, as defined in Eq. (1).

$$A(t) = R(t) + \int_0^t R(t-x)g(x)dx \tag{1}$$

R(t) is the instantaneous reliability at *t* of the system, which is defined in Eq. (2):

$$R(t) = \int_t^\infty f(x)dx \tag{2}$$

f(x) is the probability density function of a random variable *X*, which represents the system's lifetime or time to failure.

g(x) is a renewal process rate in the interval *(0,t)*, as defined in Eq. (3)

$$g(x) = f(x) + \int_0^x g(x-u)f(u)du \tag{3}$$

m(x)dx is the probability that a renewal process cycle will be completed in the time interval [*x*, *x* + *dx*]. *R(t-x)* is the probability that the system works properly for the remaining time interval *t-x*. *R(t-x)m(x)dx* is the probability of the case that a fault has occurred and that after the repair/renewal (which occurred at the instance *x*, *0* < *x* < *t)*, the system resumed functioning with no further faults. If a system is not repairable, the concept of *A(t)* is identical with that of reliability *R(t)*.

Steady-state availability (SSA) is the system availability after a long running time, where the limiting value *A(t)* tends to decrease from 1 at the initial instant, as defined in Eq. (4) and Eq. (5).

$$A = \lim_{t\to\infty} A(t) = \frac{MTTF}{MTTF + MTTR} \tag{4}$$

$$A = \lim_{t\to\infty} A(t) = \frac{\mu}{\lambda + \mu} \tag{5}$$

The *failure rate* (λ) implies the frequency of system failure is determined by the total number of failures within an item population, divided by the total time expended by that population, during a particular measurement interval under the stated conditions. *Repair rate* (μ) implies the frequency of system repair determined as the average number of repairs over a period of maintenance time. *Mean time to failure (MTTF)* represents the expected time in which a system functions correctly before its first failure. *Mean time to repair (MTTR)* represents the expected time required for system repair. In the case where failure/repair events comply with exponential distributions, *MTTF* and *MTTR* represent an arithmetic inversion of failure and repair rates, as shown in Eq. (6). SSA can be computed from Eq. (5).

$$MTTF = \frac{1}{\lambda} \quad MTTR = \frac{1}{\mu} \tag{6}$$

In industry, system administrators are usually concerned with *system downtime* (measured in minutes per year) and *downtime cost* (with a cost unit C per minute of system downtime). These values can be computed with Eq. (7) and (8).

$$Downtime = (1 - A) * 8760^{*}60 \tag{7}$$

$$Downtime\ \mathrm{Cos}\, t = C^{*}(1 - A) * 8760^{*}60 \tag{8}$$

Sensitivity analysis is performed to assess the importance of system parameters by two techniques. (*i*) Repeatedly substitute specific parameter values in one range at a time while the others remain constant, and observe system behaviors in accordance with the variation of the selected parameter. This approach studies the system responses upon a broad range of the parameters under consideration. (*ii*) *Differential sensitivity analysis*: compute partial derivatives of the measure of interest with respect to each system parameter as determined in Eq. (9) or (10) to yield a scaled sensitivity.

$$S_\tau(A) = \frac{\partial A}{\partial \tau} \tag{9}$$

$$SS_\tau(A) = \frac{\partial A}{\partial \tau}\left(\frac{\tau}{A}\right) \tag{10}$$

Stochastic reward net (SRN) [8] has been an appropriate modeling paradigm to capture operational complexities in industrial hardware and software systems [9–14]. According to a specific description of system operations, ones can model system behaviors using place(s), transition(s) and arc(s) as three main components in an SRN model. To represent a certain entity of the system to be considered, we use token(s) (normally denoted by a dot or an integer number to represent a number of corresponding entities) which reside in each place of the SRN model. And to capture its operational state variations, we use (input/output) arcs to connect transition(s) to place(s) or place(s) to transition(s), respectively. A firing of a transition is triggered when a certain condition of system state is matched in order to allow the token(s) in a place are removed, and then deposited in another place. The transitions of tokens in an SRN model captures the system's operations while the residence of tokens in places represent the system's operational state at a time, which is call marking. The Boolean condition attached to each transition which is to enable/disable the transition is called the guard. A set of guard functions can be defined to articulate the behaviors of system state dependence and transition. A marking-dependence (denoted by a # sign attached to a transition) is incorporated when the transition's rate is dependent on the marking of the SRN model at a time. Other features of SRN including inhibitor arcs, multiplicities, and input arcs can simplify the construction of SRN models.

SRN-based availability quantification framework is presented in **Figure 1**. The availability quantification framework consists of three stages: (i) requirement specification, (ii) SRN-based system modeling and (iii) system analysis. Service level agreement (SLA) [15, 16] between system owner and customer details system specification and requirements. In the stage (i), taking into

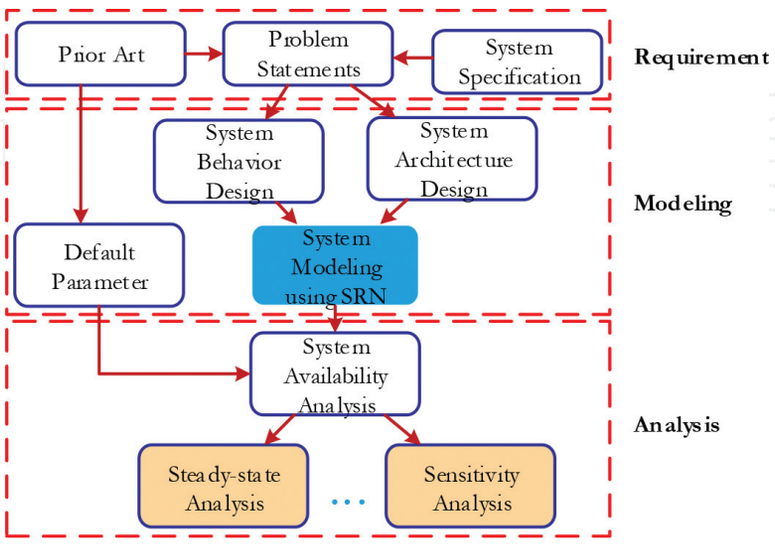


Figure 1. SRN-based availability quantification framework.

account the literature review based on prior art and contemporary development of the system, ones can define problem statements to be modeled and observed. In the stage (ii), the person in charge of modeling and evaluating the system can refer various default values of system parameters from previous work. He/she can propose the architecture design and detailed behaviors taken into consideration of the system. The SRN is used to capture the pre-defined system operations. The SRN system model is then analyzed and the system availability evaluation is performed with regard to various output measures of interest via different analysis approaches such as steady-state availability and/or sensitivity analysis.

3. Case study I: a virtualized server system

3.1. System architecture

Figure 2 shows a general VSS architecture. A VSS is a computing unit in a DC which consists of a number of physical servers (also called hosts *H1, H2, …, H_n*). Each server is in turn virtualized using bare-metal virtualization technology [17–19]. Thus, each server hosts its own hypervisor (hereinafter, called the virtual machine monitor (VMM)). The physical server is capable of running a number of virtual machines (VM) on top of its VMM. For the sake of fault tolerance and data storage of VMMs and VMs, the physical servers are interconnected via a network pipeline to each other, and to a shared storage area network (SAN).

To focus on modeling complex behaviors of a virtualized system in a detailed manner, we consider a small-size VSS consisting of two hosts (*H1* and *H2*) connected to a shared SAN. Each host runs its own virtual machine monitors *VMM1* and *VMM2*, respectively. Two VMs are also created on each host, *VM1* for host *H1* and *VM2* for host *H2*. In the next section, we will present SRN models of the above-mentioned subsystems. The models capture in detail various failure modes and recovery methods, including hardware failures in physical hosts and SAN [20, 21], failures due to non-aging related Mandelbugs on both VMM and VM subsystems [22], and software aging-related failures and corresponding time-interval software rejuvenation techniques for VMM and VM subsystems [23, 24]. Furthermore, we incorporate

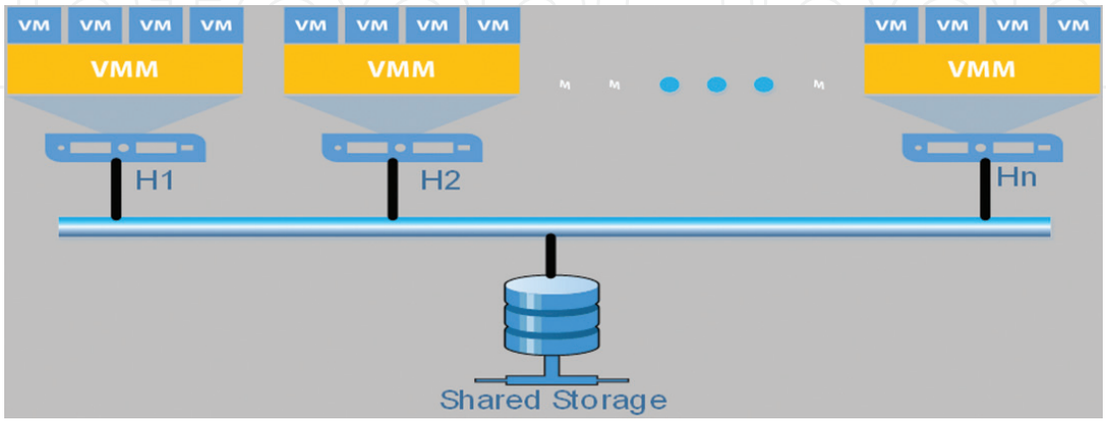


Figure 2. A virtualized server system with two physical servers.

hierarchically complex dependencies between subsystems, including the dependences of a VM on its VMM, a VM on the shared SAN, and a VMM on its host. Without loss of generality, the proposed SRN model represents the sophisticated operations of, and interactions between subsystems, in a typical virtualized system as a computing unit brick in a practical DC. The model can be further extended in the future by incorporating a large scale cloud system as in [25].

3.2. SRN models of VSS

The SRN system model is presented in **Figure 3**. We use a two-state SRN model to capture the operational state (UP) and failed state (DOWN) of the physical parts, including host 1 (*H1*), host 2 (*H2*), and SAN, as shown in **Figure 3(a)–(c)**, respectively.

The VMM subsystem models are shown in **Figure 3(d)** and **(f)** for *VMM1* and its clock, respectively, and in **Figure 3(e)** and **(g)** for *VMM2* and its clock, respectively. Without loss of generality, a model of a VMM (either *VMM1* or *VMM2*) subsystem consists of six states (represented by shaded places): (i) normally running state (P_{VMMup}), (ii) failure state due to non-Mandelbugs (P_{VMMf}), (iii) down-state due to a failure of its underlying host (P_{VMMdn}), (iv) failure-probable state due to aging problems (P_{VMMfp}), (v) aging-failure state due to aging of equipment (P_{VMMaf}), and (vi) rejuvenation-process state (P_{VMMrej}). Initially, there is a token in P_{VMMup} to represent a running VMM. If it fails due to a non-aging Mandelbug, the transition time T_{VMMf} is fired to transit the token into P_{VMMf}. Recovery is captured by $T_{VMMrepair}$. After running for a long time, the VMM suffers a high failure probability while remaining operational. Therefore, it goes to the failure-probable state P_{VMMfp} as T_{VMMfp} is fired. Failure due to aging occurs soon after T_{VMMaf} is fired and the VMM goes to the aging-failure state P_{VMMaf}. Its recovery is represented by the firing of T_{VMMar}. If the VMM's underlying host goes down (i.e., a token is deposited in P_{Hf} in respective **Figure 3(a)** or **(b)**) while the VMM is in the UP states (normal P_{VMMup} or failure-probable P_{VMMfp}), the VMM immediately enters the down-state P_{VMMdn} through the immediate fired transitions $t_{VMMupdn}$ or $t_{VMMfpdn}$. A reset is necessary for the VMM to go up (captured by $T_{VMMreset}$) after its host is recovered. In the meantime, the VMM clock is initiated by a token in $P_{VMMclock}$, which counts time by firing a timed transition $T_{VMMclockinterval}$ that complies with the c_{VMM}-stage Erlang distribution. Every software rejuvenation process interval on a VMM is represented by a firing of $T_{VMMclockinterval}$, and the token in $P_{VMMclock}$ is removed and deposited in $P_{VMMpolicy}$. Thus, rejuvenation is triggered if there is a VMM in P_{VMMup} or P_{VMMfp} by firing the immediate transitions $t_{VMMuprej}$ or t_{VMMrej}. Also, the token in $P_{VMMpolicy}$ of the VMM clock model is moved to $P_{VMMtrigger.}$ The VMM represented by a token in P_{VMMrej} is then rejuvenated and returned to the normal state P_{VMMup} as T_{VMMrej} is fired. The VMM clock is reset as $t_{VMMclockreset}$ is fired to start a new interval of time-based software rejuvenation on a VMM. The modeling of VMM1 on host H1 and VMM2 on host H2 are identical based on the general model description as above.

Modeling of VM subsystems is shown in **Figure 3(h)** and **(j)** for VM1 subsystem and its clock, respectively, and **Figure 3(i)** and **(k)** for VM2 subsystem and its clock, respectively. The models initiate with two tokens in P_{VMup} representing two VMs on each host. In general, the SRN model of a VM subsystem also consists of six states as in the VMM subsystem does including: (i) normal state (P_{VMup}), (ii) failure state due to non-aging Mandelbugs (P_{VMf}), (iii) down-state

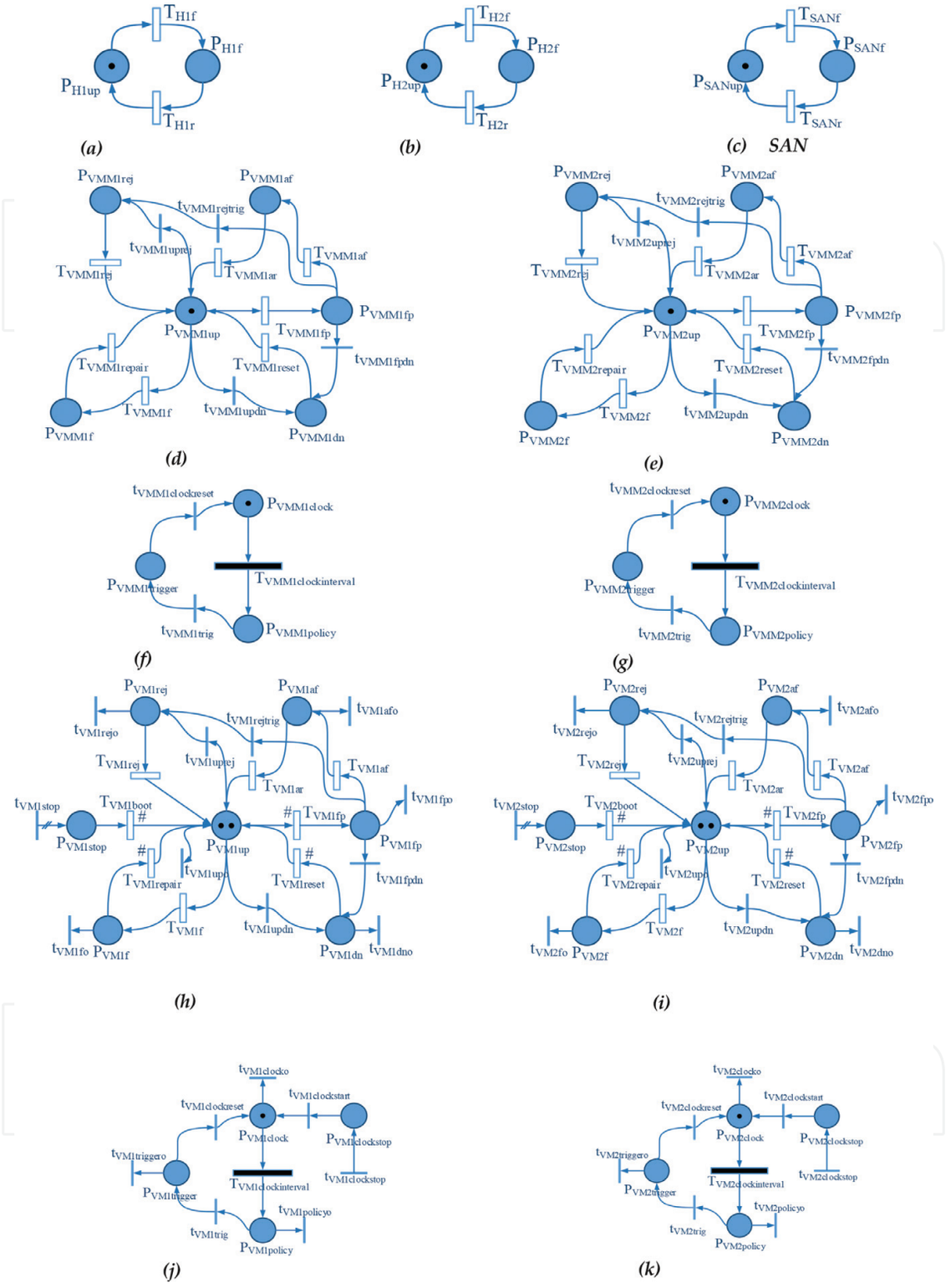


Figure 3. SRN system model of a VSS: (a) Host 1, (b) Host 2, (c) SAN, (d) VMM1, (e) VMM2, (f) VMM1's clock, (g) VMM2's clock, (h) VM1, (i) VM2, (j) VM1's clock, and (k) VM2's clock.

due to a failure of underlying VMM (P_{VMdn}), (iv) failure-probable state due to aging problems (P_{VMfp}), (v) aging-failure state due to a failure of aging (P_{VMaf}) and (vi) rejuvenation-process state (P_{VMrej}). The operations of the VM subsystem in correspondence with the transitions of

tokens in the SRN model are similarly described as those of the VMM subsystem. However, the SRN model of the VM subsystem is further extended by incorporating (i) marking-dependence represented by a "#" mark nearby selected timed transitions (T_{VMfp}, T_{VMf}, $T_{VMreset}$) to capture the cases in which two VMs in the same state compete with each other in order to transit to a new state and (ii) dependence between the VM subsystem and SAN. The second dependence is captured by the immediate transitions t_{VMupo}, t_{VMfo}, t_{VMdno}, t_{VMfpo}, t_{VMafo}, and t_{VMrejo} in the VM model, and $t_{VMclocko}$, $t_{VMpolicyo}$, and $t_{VMtriggero}$ in the VM clock model. As the SAN fails (depicted by a token in P_{SANf}), these transitions are fired to remove tokens in the VM model and VM clock model, regardless of their locations representing the loss of VM images on SAN and VM clock functionalities. Nevertheless, as soon as the SAN is recovered, two VMs are immediately created on the SAN, and they are booted onto a VMM of a corresponding host. The creation of multiple VMs is captured by t_{VMstop}, whereas the booting of a VM in the sequence is captured by T_{VMboot} with marking-dependence. The VM clock is also started after the recovery of a SAN, as captured by $P_{VMclockstop}$ and two immediate transitions $t_{VMclockstop}$ and $t_{VMclockstart}$.

3.3. Availability analysis scenarios and results

We implemented the SRN models in the Stochastic Petri Net Package (SPNP) [26]. Input parameters are selected based on previous work [20, 27], as shown in **Table 1**.

Input	Description	Transitions	Value	Input	Description	Transitions	Value
μ_{hr}	Host repair	T_{H1r}, T_{H2r}	3 days	λ_{hf}	Host fail	T_{H1f}, T_{H2f}	1 years
λ_{vmmf}	VMM non-aging failure	T_{VMM1f}, T_{VMM2f}	2654 hours	λ_{vmf}	VM non-aging failure	T_{VM1f}, T_{VM2f}	2893 hours
μ_{vmmr}	VMM reset	$T_{VMM1reset}$, $T_{VMM2reset}$	1 min	δ_{vmr}	VM repair	$T_{VM1repair}$, $T_{VM2repair}$	30 min
δ_{vmmr}	VMM repair	$T_{VMM1repair}$, $T_{VMM2repair}$	100 min	μ_{vmr}	VM restart	$T_{VM1reset}$, $T_{VM2reset}$	50s
β_{vmmfp}	VMM failure-probable	T_{VMM1fp}, T_{VMM2fp}	2 months	β_{vmfp}	VM failure-probable	T_{VM1fp}, T_{VM2fp}	1 month
λ_{vmmaf}	VMM aging-failure	T_{VMM1af}, T_{VMM2af}	2 weeks	λ_{vmaf}	VM aging failure	T_{VM1af}, T_{VM2af}	1 week
μ_{vmmar}	VMM aging recovery	T_{VMM1ar}, T_{VMM2ar}	120 min	μ_{vmar}	VM aging recovery	T_{VM1ar}, T_{VM2ar}	120 min
τ_{vmm}	VMM clock interval	$T_{VMM1clockinterval}$, $T_{VMM1clockinterval}$	1 week	τ_{vm}	VM clock interval	$T_{VM1clockinterval}$, $T_{VM2clockinterval}$	3.5 days
β_{vmmrej}	VMM rejuvenation	$T_{VMM1rej}$, $T_{VMM1rej}$	2 min	β_{vmrej}	VM rejuvenation	T_{VM1rej}, T_{VM2rej}	1 min
λ_{sf}	SAN fail	T_{SANf}	1 year	η_{vmb}	VM booting after VMM rejuvenation	$T_{VM1boot}$, $T_{VM2boot}$	50s
μ_{sr}	SAN repair	$T_{SANrepair}$	3 days				
c_{VMM}	c_{VMM}-stage Erlang distribution	x	10	c_{VM}	c_{VM}-stage Erlang distribution	X	10

Table 1. Input parameters of SRN models.

Cases	Description	SSA of VMM	SSA of VM
I	Rejuvenation is applied on all VMM and VM subsystems in both hosts.	0.999912470996	0.991769547666
II	Rejuvenation is not applied only on one of VMM subsystems in two hosts but applied on both VM subsystems in two hosts.	0.999908948744	0.991766082049
III	Rejuvenation is applied on both VMM subsystems in two hosts but not applied to only one of two VM subsystems.	0.999912470996	0.991770317258
IV	Rejuvenation is not applied on haft side of the system including VMM1 and VM1 subsystems but applied on VMM2 and VM2 subsystems.	0.999908948744	0.991766912872
V	Rejuvenation is not applied on both VMM subsystems in two hosts but applied on both VM subsystems.	0.999905284754	0.991763344539
VI	Rejuvenation is applied on both VMM subsystems in two hosts, but not applied on both VM subsystems.	0.999912470996	0.991771080172
VII	Rejuvenation is not applied on VMM and VM subsystems in both hosts.	0.999905284754	0.99176419998

Table 2. Analysis scenarios of VSS and SSAs of VMM and VM subsystems.

- *Steady-state availability:* We conducted numerical experiments in seven case studies with regard to different rejuvenation combinations. The case studies are described along with analysis results of SSA of VMM and SSA of VM in **Table 2**. The reward functions used to compute SSAs are defined as

$$SSA_{VMM} = \begin{cases} 1 : if \ \left(\#P_{VMM1up} + \#P_{VMM1fp} + \#P_{VMM2up} + \#P_{VMM2fp}\right) > 0 \\ 0 : otherwise \end{cases}$$
$$SSA_{VM} = \begin{cases} 1 : if \ \left(\#P_{VM1up} + \#P_{VM1fp} + \#P_{VM2up} + \#P_{VM2fp}\right) > 0 \\ 0 : otherwise \end{cases} \tag{11}$$

where $\#P_X$ is the number of token in place P_X. The results show that the following:

i. Time-based rejuvenation techniques with default parameters, when implemented on both VMM and VM subsystems in combination does not gain the highest SSA for the virtualized system. When a VMM undergoes a rejuvenation process, it pulls down all VMs running on top of the VMM;

ii. Rejuvenation on VMM exposes more effectiveness in gaining higher SSA in comparison to the VM.

iii. An appropriate rejuvenation combination implemented on either a VMM or VM with proper clock intervals can actually enhance system availability.

- *Sensitivity analysis of SSA*: The sensitivity analysis is observed in five case studies w.r.t the variation of: (i) only VMM1 clock's interval; (ii) only VM1 clock's interval; (iii) both VMM1 and VMM2 clocks' interval; (iv) both VM1 and VM2 clocks' interval; and (v) all clock intervals with the same duration, as shown in **Figure 4**. The findings are as follows:

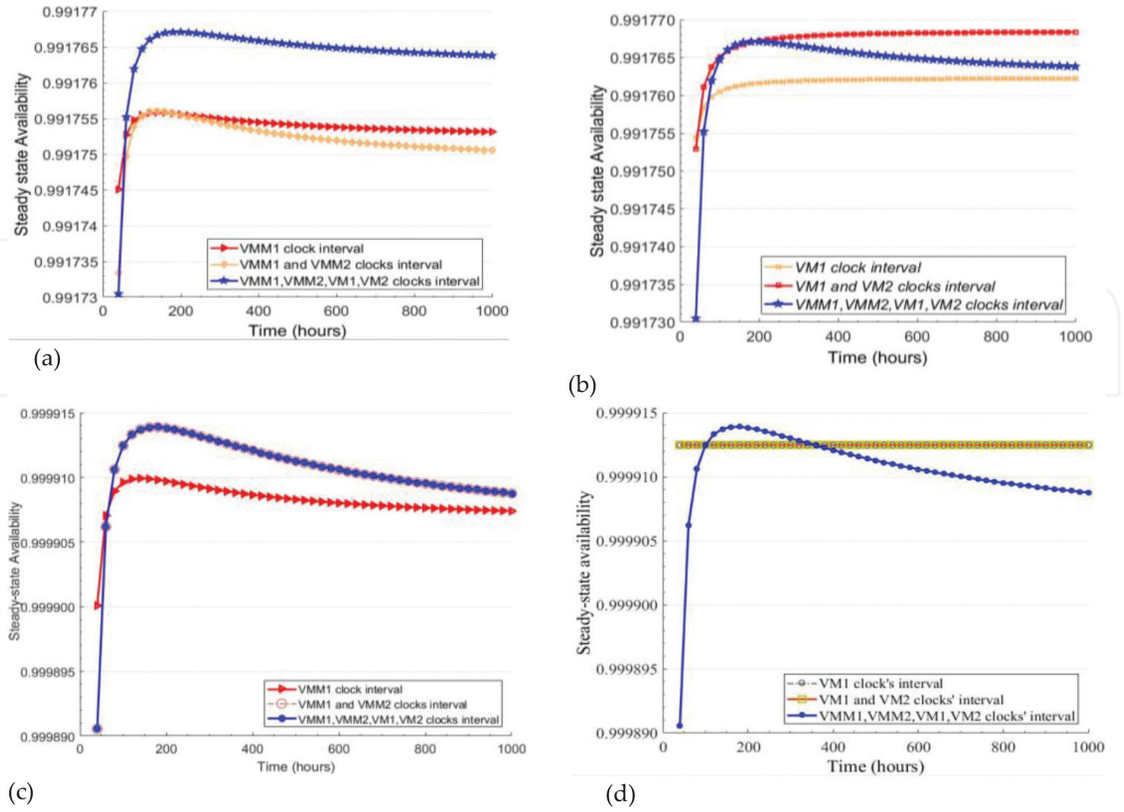


Figure 4. Sensitivity analysis of SSA of VMM and VM subsystems: (a) SSA of VM with respect to VMM clocks' intervals, (b) SSA of VM with respect to VM clocks' intervals, (c) SSA of VMM with respect to VMM clocks' intervals, and (d) SSA of VMM with respect to VM clocks' intervals.

i. **Figure 4(a)** and **(b)** shows that rejuvenation processes on VMM reduce SSA of the VM, but those on VM can improve. A proper combination of rejuvenation processes on the VMM and VM can yield an efficient impact for maintaining high values of SSA of VM.

ii. **Figure 4(c)** and **(d)** shows that there is no dependence of a VMM on its VM incorporated in the modeling of the proposed VSS yet. Also, rejuvenation implemented on both VMM subsystems of both hosts obviously gains higher SSA of VMM than it would if implemented on only one of the VMM subsystems.

4. Case study II: a DCell-based data center network

4.1. A typical DCN architecture

In this section, the DCell in consideration is expanded in size up to a network of virtualized servers complying a DCell topology. A DCell [28] is recursively constructed based on the most basic element $DCell_0$ as follows:

i. A $DCell_0$ consists of n physical servers connected to an n-port switch.

ii. A $DCell_1$ is composed of $n + 1$ $DCell_0$s. Each server of a $DCell_0$ in a $DCell_1$ has two links. One connects to its switch, the other connects to the corresponding server in another $DCell_0$, complying with a predetermined DCell routing algorithm. Consequently, every pair of $DCell_0$s in a $DCell_1$ has an exact unique link between each other.

iii. A $DCell_k$ is a level-k of $DCell_{k-1}$.

To apply the proposed modeling approach using SRN, we focus on studying a special case of DCell-based DCN at level 1 ($DCell_1$). Particularly, a cell $DCell_0$ consists of two physical servers and one shared switch. $DCell_1$ is composed of three $DCell_0$s, as shown in **Figure 5**. We assume that each server has two NICs, one for connecting to the switch in the same cell, and the other for direct connection between the server in a cell and the corresponding server in another cell, which complies with DCell network routing topology. The system architecture is detailed as follows: (i) $DCell_0$[0] consists of switch *S0*, two hosts *H00* and *H01*, a number of VMs (n_{00} of *VM00* and n_{01} of *VM01*) on the hosts *H00* and *H01*, respectively; (ii) the description of other cells goes in the same manner.

4.2. Proposed SRN model

The SRN system model of the DCell-based DCN is presented in **Figure 6**. To simplify the modeling and to focus on sophisticated interactions between VMs and servers in a cell and in different cells of the network, we use two-state SRN models (consisting of UP and DOWN states) for physical parts of the system, including hosts and switches, as shown in **Figure 6(a)–(j)**. Initially, there is a token in the UP state for each model of a certain physical part, which is

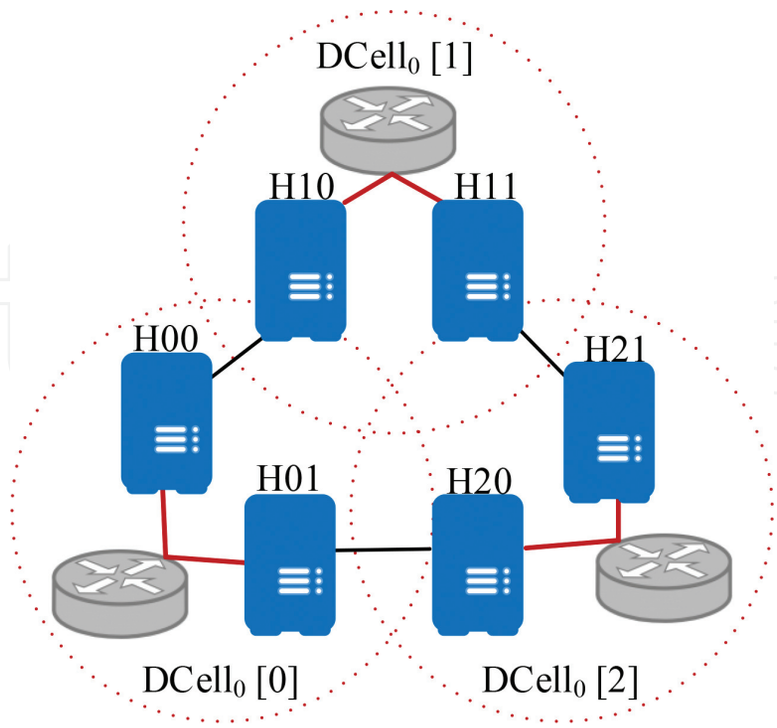


Figure 5. An architecture of a DCell-based data center network.

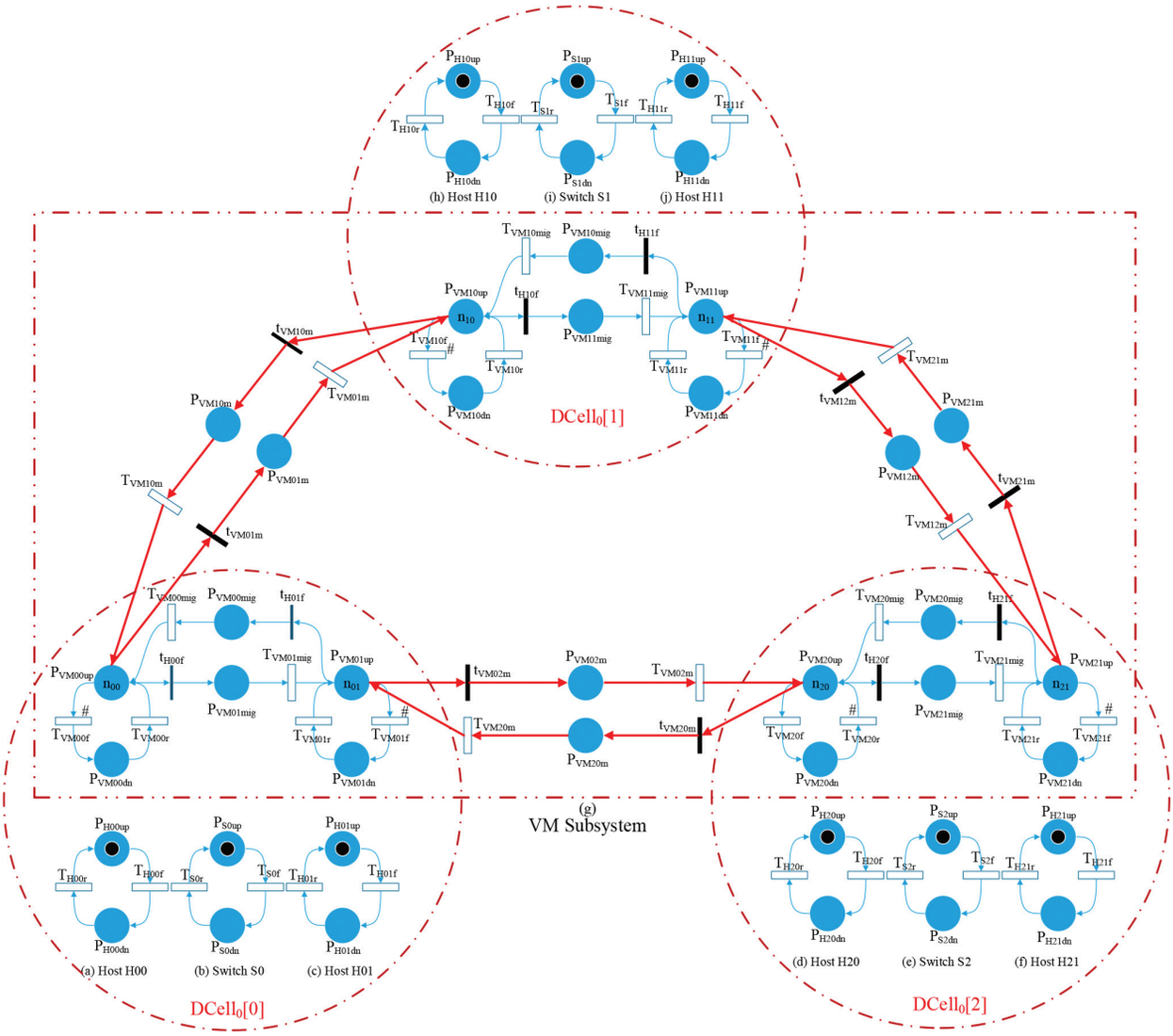


Figure 6. SRN system model of a DCell-based data center network.

depicted by a black dot which represents the initial normal working state of the physical hosts and switches. Contrary to the presented case-study of VSS in Section 3, we do not take into account the modeling of the VMM subsystem. Instead, we combine host and VMM in a unique model by considering the mean time to failure equivalent (MTTFeq) and mean time to repair equivalent (MTTReq) of the VMM subsystem as input parameters in the two-state models of hosts. Also, we simplify the modeling of the VM subsystem by using only two-state SRN models as shown in **Figure 6(g)** (VM subsystem model). There is an initial number of VMs on each host in a general case as represented by tokens in UP states. Specifically, there are n_{00} of VMs in P_{VM00up}, and n_{01} of VMs in P_{VM01up} in cell $DCell_0$[0]. In $DCell_0$[1], the numbers of VMs initially running in a normal state on each host are n_{10} of *VM10*, and n_{11} of *VM11*, which are hosted on *H10* and *H11*, respectively. Those numbers in $DCell_0$[2] are n_{20} of *VM20* and n_{21} of *VM21*. Unlike the SRN model of a single unit of VSS in **Figure 3**, we capture in the SRN system model the VM live migration techniques within a cell and between different cells for the sake of fault tolerance and improvement of system availability.

The VM migration is implemented between two hosts in a cell when a host in the cell experiences downtime due to a certain failure. In cell $DCell_0$[0] for instance, the VM live migration is triggered to migrate all running VMs from the host *H00* to the host *H01* immediately when the host *H00* fails (represented by a token in P_{H00dn}). The immediate transition t_{H00f} is triggered to remove all tokens in P_{VM00up} and deposit them in $P_{VM01mig}$. As the timed transition $T_{VM01mig}$ is fired, the tokens in $P_{VM01mig}$ are removed and deposited in P_{VM01up}, representing the completion of VM live migration processes from *H00* to *H01*. If host *H01* fails (i.e., a token is placed in P_{H01dn}), the VM live migration is performed from *H01* to *H00* and is captured by the immediate transition t_{H01f} (to trigger VM live migration processes), the place $P_{VM00mig}$ (the state of a VM in migration), and the timed transition $T_{VM00mig}$ (to represent the migration processes that take time to complete). The description of VM live migration within a cell occurs in the same manner for other cells $DCell_0$[1] and $DCell_0$[2].

In the case of a failed switch in a cell, VM live migration is performed between two hosts in two different cells via a peer-to-peer connection. For instance, if switch *S0* fails, the connections between the two hosts *H00* and *H01* in cell $DCell_0$[0] and the two host connections to outside users are disrupted. However, the number of *VM00* and *VM01* are still running on hosts *H00* and *H01*, respectively. It is necessary to migrate these VMs to other cells in order to enhance the overall availability of the system. The VM migration processes from cell $DCell_0$[0] to the other two cells are triggered by the two immediate transitions t_{VM01m} (to migrate VMs from $DCell_0$[0] to $DCell_0$[1]) and t_{VM02m} (to migrate VMs from $DCell_0$[0] to $DCell_0$[2]). After that, the tokens in P_{VM00up} are removed and deposited in P_{VM01m} and are then deposited in P_{VM10up} in cell $DCell_0$[1] as T_{VM01m} is fired. The transition of tokens P_{VM00up} in $DCell_0$[0] to P_{VM10up} in cell $DCell_0$[1] captures the migration of VM on host *H00* after a failure of switch *S0* between the two different cells. On the other side, the tokens in P_{VM01up} are removed and deposited in P_{VM02m} and are then deposited in P_{VM20up} in cell $DCell_0$[2]. This represents the migrations of VMs on host *H01* after the failure of switch *S0* from cell $DCell_0$[0] to cell $DCell_0$[2].

Without loss of generality, the VM live migration techniques within a cell and between two cells are described in detail as above for cell $DCell_0$[0]. These migrations apply similarly to the other cells $DCell_0$[1] and $DCell_0$[2].

4.3. Availability evaluation

The proposed SRN models are all implemented in SPNP. The default input parameters are listed in **Table 3**. To reduce the complexity of model analysis, we initiate only one VM on each host *H00*

Input	Description	Values	Input	Description	Values
λ_H	Host failure rate	800 hours	μ_H	Host repair rate	9.8 hours
λ_{VM}	VM failure rate	4 months	μ_{VM}	VM repair rate	30 min
λ_S	Switch failure rate	1 year	μ_S	Switch repair rate	24 hours
ω_{mig}	Network bandwidth within a $DCell_0$	1 GB/s	ω_m	Network bandwidth between two $DCell_0$s	256 Mb/s
S_{VM}	VM image size	10 GB	n_{00}, n_{01}	No. Of initial VMs in $Dcell_0$[0]	1

Table 3. Default input parameters for SRN system model of a DCN.

and *H01* in cell $DCell_0$[0] in the default case, and there are no other VMs in the other cells. However, we also evaluate the impact of the number of VMs in the DCN on the overall system availability. In this case-study, we consider two different evaluation scenarios: (I) a standalone *DCell0* (with two hosts and one switch), and (II) the proposed three-cell DCN (as modeled above). The reward rates used to compute SSA of the two cases are defined as follows:

$$A_I = \begin{cases} 1 : if \ \left(\#P_{VM00up} + \#P_{VM01up} > 0\right)\&\&\left(\#P_{S0up} == 1\right) \\ 0 : \text{otherwise} \end{cases}$$

$$A_{II} = \begin{cases} 1 : if \left\{\left(\#P_{VM00up} + \#P_{VM01up} > 0\right)\&\&\left(\#P_{S0up} == 1\right)\right\} \\ \| \left\{\left(\#P_{VM10up} + \#P_{VM11up} > 0\right)\&\&\left(\#P_{S1up} == 1\right)\right\} \\ \| \left\{\left(\#P_{VM20up} + \#P_{VM21up} > 0\right)\&\&\left(\#P_{S2up} == 1\right)\right\} \\ 0 : \text{otherwise} \end{cases} \tag{12}$$

- Steady-state availability:
  - We first evaluate SSA and downtime of the two scenarios as shown in **Table 4**. We assume that a minute of system downtime incurs a penalty of 16,000 USD for the system owner according to the SLA *signed* with customers [29]. The results clearly show that the proposed three-cell DCN obtains much higher availability, and thus reduce downtime minutes and downtime cost penalty in a year than a standalone cell with only two physical servers.
  - We also evaluate the impact of the initial number of VMs in a DCN on the system's overall availability, as shown in **Table 5**. The results show that as we increase the initial number of VMs, the overall system availability also increases. The increased SSA in the proposed three-cell DCN is also faster than in the standalone $DCell_0$. However, if the initial number of VMs (represented by the total number of tokens in the proposed SRN system model) obtains a large value, it causes a *memory error* in computing the system availability due to the largeness problem of the SRN model.
- *Sensitivity analysis of SSA:* We observe the variation of SSA in accordance with changes in the selected input parameters, including MTTF and MTTR of hosts, VMs and switches, and VM migration rate between two hosts in a cell or in two different cells, as shown in **Figure 7**. The results show that:

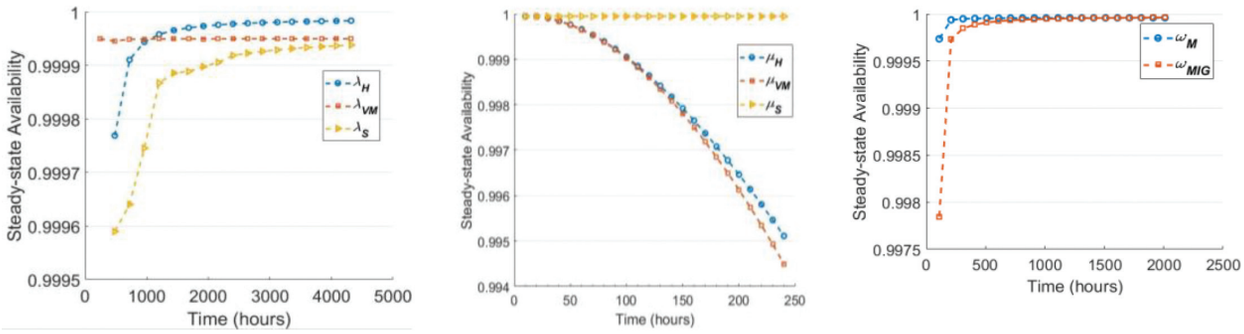
Case	Description	SSA	No. of nines	Downtime (min/year)	Downtime cost (USD/year)
I	Standalone $DCell_0$	0.997240422469	2.55	1450.4	23,206,943
II	Proposed three-cell DCN	0.999950276761	4.30	26.1	418,152

Table 4. Steady-state availability and downtime cost.

n_{VM}	I		II	
	SSA	**#nines**	**SSA**	**#nines**
1	0.997064755072	2.532356	0.999773875854	3.646
2	0.997240422469	2.559157	0.999950276761	4.303
3	0.997240488479	2.559168	0.999950574780	4.306
4	0.997240519634	2.559173	0.999950839446	4.308
5	0.997240550678	2.559178	0.999951101800	4.311
6	0.997240759564	2.559210	*m.e*	*m.e*

(m.e: memory error)

Table 5. Impact of number of VMs.



(a) (b) (c)

Figure 7. Sensitivity analysis with respect to impacting parameters. (a) MTTF, (b) MTTR, (c) VM migration rate.

- SSA is improved as we increase MTTFs and VM migration rates, and as we decrease MTTRs.
- In **Figure 7(a)**, we see that the switch is an important component of the network because its MTTF is small. Thus, the SSA clearly drops down vertically in comparison to the MTTFs of other components. Furthermore, MTTF of a host is a significant parameter in the long-run since it causes a better enhancement in the overall availability than the other MTTFs.
- In **Figure 7(b)**, we clearly find that the repair time of a switch does not affect the SSA because we perform VM migration between cells to tolerate the failures of switches. This ensures that VMs can be migrated to other cells, regardless of the failure/recovery of a certain switch. However, we can see that the recovery of a VM has a greater impact on SSA than that of a host.
- In **Figure 7(c)**, the migration rates of VMs between cells can clearly enhance SSA in comparison with those within a cell. However, the low value of the VM migration rate within a cell severely drops the system's availability.

5. Case study III: a Disaster Tolerant Data Center (DTDC)

5.1. A typical system architecture of a DTDC

This case-study considers disaster tolerance of cloud computing in a DCS. The system is composed of two different DCs (DC1 and DC2), which are geographically located in two distant regions, as shown in **Figure 8**. In each DC, we place a VSS of two physical servers (H1 and H2 in DC1, and H3 and H4 in DC2). All physical machines are assumed to be identical. Each server is initially capable of running a VM (VM1~VM4 runs on H1~H4, respectively). Shared network attached storage (NAS) is equipped in each DC to provide distributed storage and a VM migration mechanism between two hosts in the same DC. To implement disaster tolerance and recovery strategies between DCs, a back-up server is incorporated to provide VM data backup. The back-up server allows periodic synchronization of VM data between DCs. This allows the most-updated VM data to be recovered onto an operational DC after a disaster strikes on another DC.

Furthermore, to enhance the system's overall availability, we use the (active-standby) fail-over technique and VM switching mechanism. Specifically, when a VM on a certain host fails, a standby VM on the same host wakes up and takes over the operations of the failed VM. If there is no standby VM on the same host, the standby VM on the remaining host goes up and takes place on the failed host.

If a host in a DC fails, its VMs in the standby state are switched on in order to load onto the remaining host. Various VM migration mechanisms are also taken into account in this system. VM live-migration is performed between two hosts in a DC when one of the hosts fails. VM migration between two DCs is triggered when a DC undergoes a system failure when two hosts enter a downtime period simultaneously. When a disaster devastates a DC, VM migration between the back-up server (in a safe zone) and the remaining operational DC is implemented as a means of disaster recovery.

5.2. Availability modeling of a DTDC

The SRN system model for availability quantification of the studied DTDC is shown in **Figure 9**. We use simplified two-state SRN models (UP and DOWN) to capture general failure and recovery behaviors of physical parts in the system, including the physical hosts H1–H4 (**Figure 9(a)**, **(b)**, **(j)**,

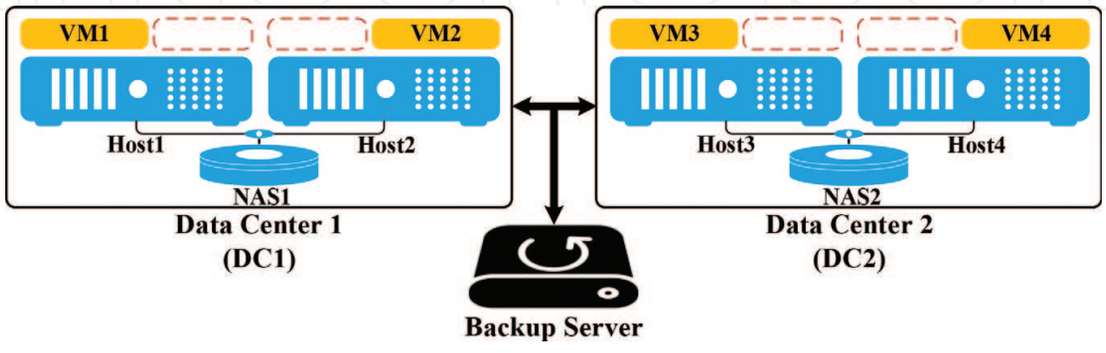


Figure 8. A conceptual architecture of a disaster tolerant data center system.

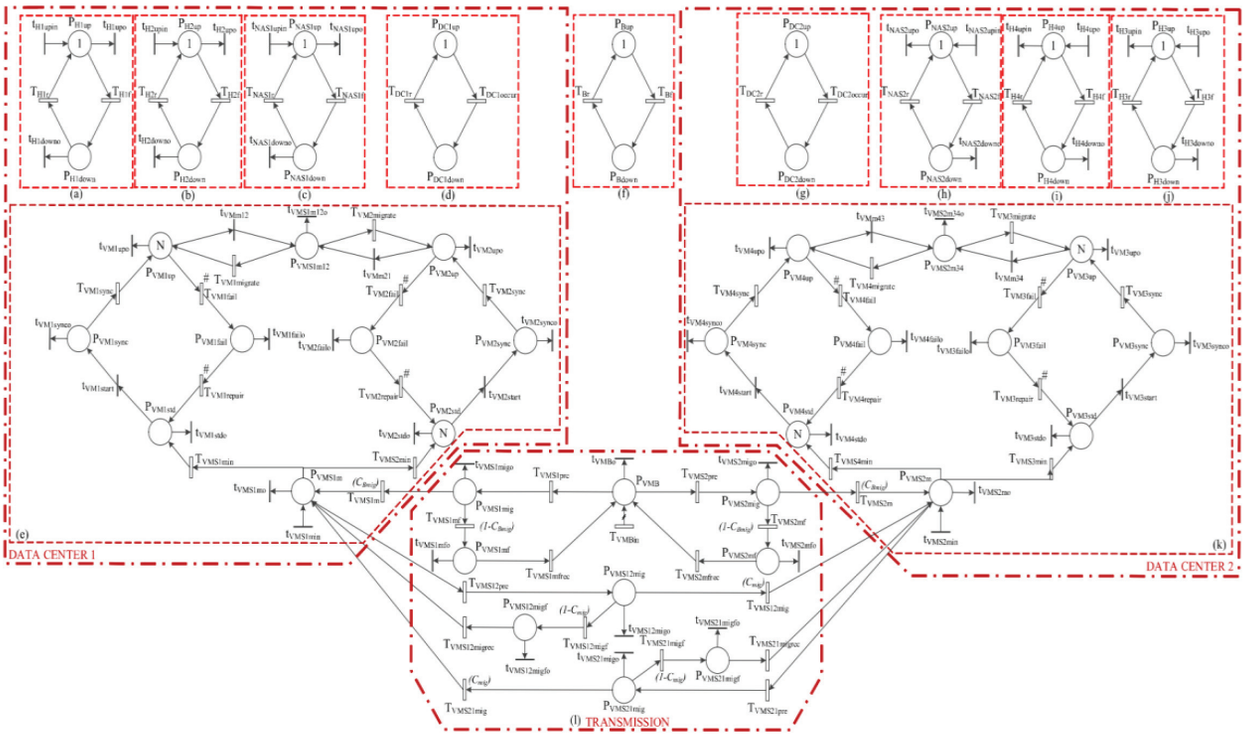


Figure 9. SRN system model of a disaster tolerant data center.

and **(i)**, respectively), NAS1 in DC1, and NAS2 in DC2 (**Figure 9(c)** and **(h)**, respectively). We use immediate transitions t_{Hupo}, t_{Hdowno}, t_{NASupo}, and $t_{NASdowno}$ to remove tokens in the up and down places of the host and NAS models in order to represent the entire operational termination of a DC when a disaster strikes. When the disaster passes and the reconstructed DC starts a new operational cycle, the immediate transitions t_{Hupin} and $t_{NASupin}$ are used to deposit new tokens in the up states of the host and NAS models. The occurrence of a disaster at a site is also represented by using a two-state model as shown in **Figure 9(d)** and **(g)** for the occurrence of a disaster at DC1 and DC2, respectively. The two-state SRN model in **Figure 9(f)** captures the operational and failure states of the back-up server.

The modeling of VM subsystems in DC1 and DC2 are shown in **Figure 9(e)** and **(k)**, respectively. Since we initially assume that all hosts and VMs are identical, the modeling of the two DCs is also identical. The model initializes N tokens in P_{VM1up}, and the other N tokens in P_{VM2std} represent N operational VMs with their N standby VMs at the beginning. Each VM sub-model mainly has four states, including the operational state (P_{VMup}), failure state (P_{VMfail}), standby state (P_{VMstd}), and synchronization state (P_{VMsync}). If a VM fails, it moves from the upstate P_{VMup} to the failure state P_{VMfail}. When the failed VM is repaired, it moves to the standby state P_{VMstd}. At this point, the active-standby fail-over mechanism of VMs is captured as follows. When a VM fails, a standby VM (represented by a token in P_{VMstd}) on the same host (before the disaster) or on the remaining host (after the disaster) transits to P_{VMsync} in order to synchronize the most-updated data on the NAS of that DC corresponding to the previously failed VM. It then goes up to P_{VMup} and takes the place of the failed VM. Dependence marks are placed near timed transitions T_{VMfail} and $T_{VMrepair}$ to represent the competition between

failure and repair of VMs on the same host. The VM live-migration technique is triggered as a host fails, which is captured by an immediate transition t_{VMm}, a place P_{VMSm}, and a timed transition $T_{VMmigrate}$. For instance, when host H1 fails, the VM live-migration is triggered to migrate running VMs from the failed H1 to the running H2. Thus, t_{VMm12} is triggered to fire. A number of tokens in P_{VM1up} are removed and deposited at $P_{VMS1m12}$ as it waits for migration. The timed transition $T_{VM2migrate}$ is then fired to depict the migration process of VMs onto host H2. The tokens in $P_{VMS1m12}$ are removed and deposited in P_{VM2up}. The reversed migration from host H2 to H1 is captured by t_{VMm21}, $P_{VMS1m12}$, and $T_{VM1migrate}$ in the same manner. The places P_{VMS1m} and P_{VMS2m} represent the storage of VMs on NAS1 and NAS2. When the two hosts in a DC enter downtime, all tokens in the VM sub-models of VM1 and VM2 are removed by immediate transitions t_{VMupo}, $t_{VMfailo}$, t_{VMstdo}, and $t_{VMsynco}$ (attached to four main states of VM sub-models) and deposited in P_{VMS1m} via $t_{VMS1min}$. However, if a disaster strikes, the all tokens are removed from the places in the VM sub-models via the out-going immediate transitions t_{VMupo}, $t_{VMfailo}$, t_{VMstdo}, $t_{VMsynco}$, t_{VMSmo}, and t_{VMSmo}. As the failed data center is reconstructed, a pre-defined number of VMs are created on the NAS, which is captured by depositing tokens in P_{VMSm} via t_{VMSmin}. The VMs are then assigned to hosts via the time transition T_{VMSmin}.

The VM migration techniques between the two DCs, and between the backup server and the two DCs, are modeled in **Figure 9(l)**. The place P_{VMB} represents the storage of VMs in the back-up server. When a DC is destroyed due to a disaster, its VMs are stored in the back-up server and represented by creating new tokens in P_{VMB} via the timed transition T_{VMBin}. When there is a remaining DC in its operational state, the tokens in P_{VMB} are transmitted to the corresponding P_{VMSmig} via the timed transition T_{VMSpre}. The tokens are then deposited in P_{VMSm} via the timed transition T_{VMSm} of the respective DC model with an imperfect coverage factor C_{Bmig}. If this process fails with coverage factor (1-C_{Bmig}), the tokens are moved to P_{VMS2mf} via T_{VMSmf} and returned to P_{VMB} via $T_{VMSmfrec}$. This transition of tokens captures the VM migration from the back-up server to the operational DC. In the case when the back-up server fails, the immediate transitions t_{VMBo}, $t_{VMSmigo}$, and t_{VMSmfo} remove all tokens in P_{VMB}, P_{VMSmig}, and P_{VMSmf} to represent the loss of VM image files on the back-up server. The VMs will be created on the back-up server as soon as it is recovered. The VM migration between two DCs is triggered when two hosts in a DC enter downtime simultaneously. In this case, we propose the two hosts H1 and H2 in DC1 also stay in a downtime period simultaneously. A number of VMs on DC1 are still stored in NAS1, represented by tokens in P_{VMS1m}. Thus, it is necessary to migrate these VMs onto the running DC2. The tokens are then transmitted to $P_{VMS12mig}$ after a pre-migration process ($T_{VMS12pre}$). The VM migration process is finalized with an imperfect coverage factor C_{mig} as the transition $T_{VMS12mig}$ is fired. If this migration process fails with coverage factor (1-C_{mig}), the tokens are moved to $P_{VMS12migfo}$ and returned to NAS1 in the original DC1 via $T_{VMS12migrec}$. The VM migration from DC2 to DC1 is performed similarly and captured by the places $P_{VMS21mig}$, $P_{VMS21migf}$, the timed transition $T_{VMS21pre}$, $T_{VMS21mig}$ (with imperfect coverage factor C_{mig}), $T_{VMS21migf}$ (with coverage factor 1-C_{mig}), and $T_{VMS21migrec}$.

5.3. Availability evaluation

The SRN system model is implemented in SPNP. Default input parameter values are shown in **Table 6**. We assume that the number of VMs on a host is only one in order to reduce complexity in model computation and analysis.

Input	Description	Assigned transitions	Values
λ_{Hf}	Host failure rate	T_{H1f}, T_{H2f}, T_{H3f}, T_{H4f}	800 hours
μ_{Hr}	Host recovery rate	T_{H1r}, T_{H2r}, T_{H3r}, T_{H4r}	9.8 hours
λ_{NASf}	NAS failure rate	T_{NAS1f}, T_{NAS2f}	45 years
μ_{NASr}	NAS recovery rate	T_{NAS1r}, T_{NAS2r}	4 hours
$\lambda_{DCoccur}$	Time to disaster occurrence at a DC	$T_{DC1occur}$, $T_{DC2occur}$	100 years
μ_{DCr}	DC recovery rate after a disaster	T_{DC1r}, T_{DC2r}	1 year
λ_{Bf}	Backup DC failure rate	T_{Bf}	50,000 hours
μ_{Br}	Backup DC recovery rate	T_{Br}	30 min
λ_{VMfail}	VM failure rate	$T_{VM1fail}$, $T_{VM2fail}$, $T_{VM3fail}$, $T_{VM4fail}$	4 months
$\mu_{VMrepair}$	VM repair rate	$T_{VM1repair}$, $T_{VM2repair}$, $T_{VM3repair}$, $T_{VM4repair}$	30 min
δ_{VMsync}	VM synchronization rate	$T_{VM1sync}$, $T_{VM2sync}$, $T_{VM3sync}$, $T_{VM4sync}$	5 min
$\omega_{VMmigrate}$	VM migration rate between hosts	$T_{VM1migrate}$, $T_{VM2migrate}$, $T_{VM3migrate}$, $T_{VM4migrate}$	5s
γ_{VMSmin}	VM loading rate into a host	$T_{VMS1min1}$, $T_{VMS1min2}$, $T_{VMS2min3}$, $T_{VMS2min4}$	1 s
η_{VMSpre}	VM pre-migration rate between DCs and backup server	$T_{VMS12pre}$, $T_{VMS21pre}$, $T_{VMS1pre}$, $T_{VMS2pre}$	5 min
$\theta_{VMSmigrec}$	VM return rate to NAS after a migration failure	$T_{VMS12migrec}$, $T_{VMS21migrec}$	1 min
$\theta_{VMSmfsync}$	VM synchronization rate with backup DC after a migration failure	$T_{VMS1mfsync}$, $T_{VMS2mfsync}$	1 min
C_{Bmig}	Imperfect factor of VM migration from backup DC		0.95
C_{mig}	Imperfect factor of VM migration between DCs		0.85
N	Number of VMs in a host		1
S_{VM}	Size of VM image and related data		4GB
ω_{NET}	Network speed		20 MB/s

Table 6. Default input parameters.

- *Steady state availability*: We evaluate the availability of the DTDC in seven operational scenarios by varying imperfect VM migration coverage factors between the backup server and the DCs and disaster occurrence frequency as follows: (I) The system of two standalone DCs without DT confronts disasters at the mean time to occurrence of 100 years (default value); (II) The system with default parameters; (III-V) The network connection has a high probability of failure (i.e., low probability of success in VM migration processes) and the system is planted in an area with mean disaster time set alternatively to 100, 200, and 300 years; (VI-VIII) In contrast to cases (III)-(V), the migration between distant parts may succeed with high probability and the DCs location experiences disasters with mean time to occurrence also set to 100, 200, and 300 years. The results of SSA and downtime evaluation are shown in **Table 7** such that following criteria are satisfied:

Case	C_{Bmig}	C_{mig}	$\lambda_{DCoccur}$	SSA	No. of nines	Downtime (min/year)	Downtime cost (USD/year)
I	x	X	100 years	0.989455392105	1.98	5542.2	8,675,934.6
II	0.95	0.85	100 years	0.999843164703	3.80	82.4	1,318,922.1
III	0.1	0.1	100 years	0.998942162067	2.98	556.0	8,895,993.9
IV	0.1	0.1	200 years	0.999635096345	3.44	191.8	3,068,693.8
V	0.1	0.1	300 years	0.999795681447	3.69	107.4	1,718,237.3
VI	0.9	0.9	100 years	0.999841085616	3.80	83.5	1,336,406.4
VII	0.9	0.9	200 years	0.999946639371	4.27	28.0	448,741.5
VIII	0.9	0.9	300 years	0.999968676113	4.50	16.5	263,421.4

Table 7. SSA and downtime analyses.

- The safer DCs locations (longer frequency of disaster occurrence) results in a higher system SSA.
- DCs should be placed in isolated areas to avoid any severe damage from disastrous events, even though the network connection between distant parts of the system might deal with more failure during VM migration processes.
- Higher SSA values are obtained with more reliable network connections, i.e. for network connections that can guarantee a higher success rate for transmission between distant parts of the system.

- *Sensitivity analysis:* As shown in **Figure 10**, we analyzed the sensitivity of the system's SSA with respect to different parameters, including imperfect coverage factors of VM migration (C_{Bmig} and C_{mig}), time to disaster occurrences ($\lambda_{DCoccur}$), VM image size (S_{VM}), and network bandwidth (ω_{NET}). The impact of S_{VM} and ω_{NET} is shown in **Figure 10(f)**. The

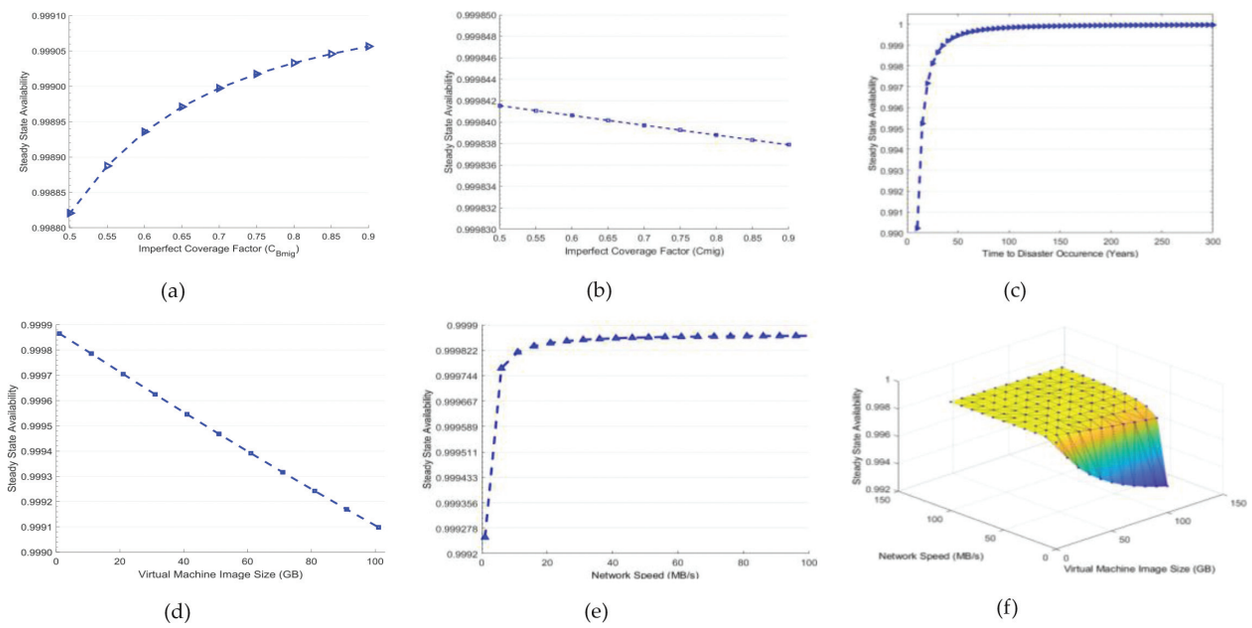


Figure 10. Sensitivity analysis of a DTDC steady state availability: (a) C_{Bmig}, (b) C_{mig}, (c) $\lambda_{DCoccur}$, (d) S_{VM}, (e) ω_{NET}, (f) ω_{NET}, S_{VM}.

results show that: (i) the disaster tolerance solution with a back-up center would improve SSA, even when connections between the back-up center with DCs incur imperfections in VM migration processes; (ii) imperfections in the VM migration processes between DCs slightly impact SSA when it increases; (iii) the system's SSA is improved vastly if DCs are located in safe areas with lower disaster occurrence frequency; (iv) larger VMs can reduce the overall availability of the system; (v) a faster network connection between distant locations can actually boost the system's availability, especially for network speeds ranging in 0-20 Mb/s, if the speed increases much higher, the effect is not much different from the default parameters; (vi) the variation of both ($\omega_{NET,}$ S_{VM}) confirms the fact that higher network speed and smaller VM sizes result in apparently higher SSA, whereas slower network and larger VMs severely reduce the system's availability.

6. Conclusion(s)

This chapter presented a set of availability models based on stochastic reward net for comprehensive system availability evaluation in data center systems. The data center systems scale during evaluation was increased from a system of two virtualized servers (considered as a unit block in data centers) in Section 3, to a typical network of virtualized servers complying with a DCell topology in Section 4. Finally, the evaluated data centers are scaled up to a two-site data center for disaster tolerance with a back-up center. A variety of fault and disaster tolerant techniques were incorporated in the systems in order to achieve high availability. The systems were evaluated under various case studies with regards to different metrics of interest, including steady state availability and its sensitivity with respect to a number of impac factors. The analysis results show comprehensive system behaviors and improved availability in accordance with incorporated techniques in the data center systems.

Acknowledgements

This research was supported by the Ministry of Science, ICT (MSIT), Korea, under the Information Technology Research Center (ITRC) support program (IITP-2018-2016-0-00465) supervised by the Institute for Information & communications Technology Promotion (IITP).

Author details

Tuan Anh Nguyen[1,2]*, Dugki Min[1] and Eunmi Choi[3]

*Address all correspondence to: anhnt2407@gmail.com

1 Office of Research, University-Industry Cooperation Foundation, Konkuk University, Seoul, South Korea

2 Department of Computer Engineering, Konkuk University, Seoul, South Korea

3 School of Management Information Systems, Kookmin University, Seoul, South Korea

References

[1] P. Insitute. 2016 Cost of Data Center Outages. Ponemon Inst; 2016

[2] Sony M, Mariappan V, Kamat V. Stochastic modelling of failure interaction: Markov model versus discrete event simulation. International Journal of Advanced Operations Management. 2011;**3**(1):1

[3] Szczerbicka H, Trivedi KS, Choudhary PK. Discrete event simulation with application to computer communication systems performance. In: Reis R, editor. Information Technology. Boston: Kluwer Academic Publishers; 2004. pp. 271-304

[4] Trivedi KS, Kim DS, Roy A, Medhi D. Dependability and security models. In: Proc. 2009 7th Int. Work. Des. Reliab. Commun. Networks, DRCN 2009; Oct. 2009. pp. 11-20

[5] Han S, Nashville T. Multidisciplinary System Reliability Analysis. NASA Contract Report. NASA CR-210969. Jun 2001. Available from: http://gltrs.grc.nasa.gov/reports/2001/CR-2001-210969.pdf

[6] Cao Y, Sun H, Trivedi KS, Han JJ. System availability with non-exponentially distributed outages. IEEE Transactions on Reliability. Jun 2002;**51**(2):193-198

[7] Trivedi KS, Kim DS, Ghosh R. System availability assessment using stochastic models. Applied Stochastic Models in Business and Industry. Mar 2013;**29**(2):94-109

[8] Nguyen TA, Min D, Choi E. A comprehensive evaluation of availability and operational cost for a virtualized server system using stochastic reward nets. The Journal of Supercomputing. Aug 2017:1-55

[9] Han K, Nguyen TA, Min D, Choi EM. An evaluation of availability, reliability and power consumption for a SDN infrastructure using stochastic reward net. In: Park JH, Pan Y, Yi G, Loia V, editors. Advances in Computer Science and Ubiquitous Computing: CSA-CUTE 2016; Singapore: Springer Singapore. 2017. pp. 637-648

[10] Raei H, Yazdani N. Performability analysis of cloudlet in mobile cloud computing. Inf. Sci. (Ny). 2017

[11] Nguyen TA, Eom T, An S, Park JS, Hong JB, Kim DS. Availability modeling and analysis for software defined networks. In: 2015 IEEE 21st Pacific Rim International Symposium on Dependable Computing (PRDC); 2015 April. pp. 159-168

[12] Dantas J, Matos R, Araujo J, Maciel P. Eucalyptus-based private clouds: Availability modeling and comparison to the cost of a public cloud. Computing. Nov 2015;**97**(11): 1121-1140

[13] Andrade E, Nogueira B, Matos R, Callou G, Maciel P. Availability modeling and analysis of a disaster-recovery-as-a-service solution. Computing. Feb 2017:1-26

[14] Raei H, Yazdani N, Shojaee R. Modeling and performance analysis of cloudlet in mobile cloud computing. Performance Evaluation. 2017;**107**:34-53

[15] Patel P, Ranabahu A, Sheth A. Service Level Agreement in Cloud Computing. Kno.e.sis Publications. The Ohio Center of Excellence in Knowledge Enabled Computing (Kno.e.sis). 2009. Available from: http://corescholar.libraries.wright.edu/knoesis/78

[16] Garg SK, Toosi AN, Gopalaiyengar SK, Buyya R. SLA-based virtual machine management for heterogeneous workloads in a cloud datacenter. Journal of Network and Computer Applications. Aug 2014;**45**:108-120

[17] Nanda S, Chiueh T. A Survey of Virtualization Technologies. SUNY; 2005

[18] Daniels J. Server virtualization architecture and implementation. Crossroads. Sep. 2009; **16**(1):8-12

[19] Ameen RY, Hamo AY. Survey of server virtualization. International Journal of Computer Science and Information Security. Apr 2013;**11**(3):65-74

[20] Kim DS, Machida F, Trivedi KS. Availability modeling and analysis of a virtualized system. In: 2009 15th IEEE Pacific Rim International Symposium on Dependable Computing, PRDC 2009; **2009**

[21] Smith WE, Trivedi KS, Tomek LA, Ackaret J. Availability analysis of blade server systems. IBM Systems Journal. 2008;**47**(4):621-640

[22] Grottke M, Nikora AP, Trivedi KS. An empirical investigation of fault types in space mission system software. In: Proceedings of 2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN); 2010. pp. 447-456

[23] Machida F, Xiang J, Tadano K, Maeno Y. Combined server rejuvenation in a virtualized data center. In: 2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing; 2012. pp. 486-493

[24] Cui L, Li B, Li J, Hardy J, Liu L. Software aging in virtualized environments: Detection and prediction. In: Proceedings of 2012 IEEE 18th International Conference on Parallel and Distributed Systems; 2012. pp. 718-719

[25] Longo F, Ghosh R, Naik VK, Trivedi KS. A scalable availability model for Infrastructure-as-a-Service cloud. In: Proceedings of 2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN); 2011. pp. 335-346

[26] Ciardo G, Muppala J, Trivedi KS. SPNP: Stochastic petri net package. In: Proc. Third Int. Work. Petri Nets Perform. Model. PNPM89; 1989. pp. 142-151

[27] Machida F, Kim DS, Trivedi KS. Modeling and analysis of software rejuvenation in a server virtualized system with live VM migration. Performance Evaluation. 2013;**70**(3):212-230

[28] Guo C, Wu H, Tan K, Shi L, Zhang Y, Lu S. Dcell: A scalable and fault-tolerant network structure for data centers. In: Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication—SIGCOMM '08. Vol. 38(4). 2008. p. 75

[29] Stansberry M. 2013 Data Center Industry Survey. Uptime Institute, LLC; 2013

IntechOpen