Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 521289, 20 pages
http://dx.doi.org/10.1155/2015/521289

Research Article

A Comprehensive Sensitivity Analysis of a Data Center Network with Server Virtualization for Business Continuity

Tuan Anh Nguyen,[1,2] Dugki Min,[1] and Jong Sou Park[2]

[1]*Department of Computer, Information & Communications Engineering, Konkuk University, 120 Neungdong-ro, Gwangjin-gu, Seoul 05029, Republic of Korea*
[2]*Department of Computer Engineering, Korea Aerospace University, 76 Hanggongdaehang-ro, Deogyang-gu, Goyang-si, Gyeonggi-do 412-791, Republic of Korea*

Correspondence should be addressed to Tuan Anh Nguyen; anhnt2407@gmail.com

Received 8 April 2015; Revised 21 August 2015; Accepted 30 August 2015

Academic Editor: Supeng Leng

Sensitivity assessment of availability for data center networks (DCNs) is of paramount importance in design and management of cloud computing based businesses. Previous work has presented a performance modeling and analysis of a fat-tree based DCN using queuing theory. In this paper, we present a comprehensive availability modeling and sensitivity analysis of a DCell-based DCN with server virtualization for business continuity using stochastic reward nets (SRN). We use SRN in modeling to capture complex behaviors and dependencies of the system in detail. The models take into account (i) two DCell configurations, respectively, composed of two and three physical hosts in a $DCell_0$ unit, (ii) failure modes and corresponding recovery behaviors of hosts, switches, and VMs, and VM live migration mechanism within and between $DCell_0$s, and (iii) dependencies between subsystems (e.g., between a host and VMs and between switches and VMs in the same $DCell_0$). The constructed SRN models are analyzed in detail with regard to various metrics of interest to investigate system's characteristics. A comprehensive sensitivity analysis of system availability is carried out in consideration of the major impacting parameters in order to observe the system's complicated behaviors and find the bottlenecks of system availability. The analysis results show the availability improvement, capability of fault tolerance, and business continuity of the DCNs complying with DCell network topology. This study provides a basis of designing and management of DCNs for business continuity.

1. Introduction

Cloud computing based businesses have been demanding a rapid escalation of IT infrastructures with efficient resources organization and high level of continuity. To endure business continuity, data centers have drastically evolved in their size and architecture design to host a variety of cloud computing applications and services such as online social networking, e-commerce services, scientific computing, and big data processing. Nevertheless, a data center becomes a centric single point of failure in the cloud infrastructure in the way that the failures of components (e.g., links, switches, and servers) may result in the overall failures of a set of connected components [1]. Internet enterprises may incur millions of dollar per hour [2] due to their service outage, since their business operations require constantly connected and online services. It is demanding to avoid such risks and improve safety of DCN. And thus designing of a DCN for fault tolerance and business continuity is critical and a sharp focus of both academia and industry areas.

Recent work has attempted to design and organize a data center's resources in the networking manner in which a large number of physical hosts are interconnected in a specific network topology called data center network (DCN), for instance, fat-tree [3], DCell [4], and BCube [5]. Thus, DCN topologies are the communication backbone in a data center [6]. The critical requirements in designing a DCN are scalability and efficiency to connect tens or even hundreds of thousands physical hosts [3, 4]. In the perspective of system users, the metrics of interest for a DCN, however, are

the overall system availability and continuity of their hosted services and applications [7]. In this context, DCell proposed by Guo et al. [4] has emerged as an appropriate solution for DCN architecture in which the architecture is extremely scalable to millions of servers in data centers [8] by recursively constructing higher level DCells based on a $DCell_0$ as the fundamental building block. The DCell network architecture allows avoiding any single point of failure and thus is able to tolerate different types of failures such as node failures, link failures, and network device failures. Furthermore, to enhance system availability and capability of fault tolerance, one may employ server virtualization [7, 9–11] into a DCN. The approach creates virtual computing machines (VMs) on each physical host of the DCN. Along with the nature of the DCell topology, the VMs become the core elements of the network to deliver high availability and fault tolerance in the way that a VM is able to be migrated from a host to another host [12, 13] and from a $DCell_0$ to another $DCell_0$ in the DCN [14] in order to avoid any hardware failures and thus to assure business continuity of system users. The DCell-based DCN with server virtualization is our sharp focus in this paper.

There are a number of papers on presentation and description of DCN topologies [3–5]. Some other work concerned with different aspects of DCN including fault tolerance characteristics [1, 15], structural robustness of DCN topologies [16], or connectivity of DCNs [6]. Nevertheless, none of these papers presented a quantitative assessment of system behaviors using stochastic models [17]. One of the previous works [18] attempted to model and analyze a simple configuration of a two-computer network with redundancy of network devices/links for fault tolerance. To the best of our knowledge, only a recent paper [19] delivered a thorough performance modeling and analysis of a fat-tree based DCN using queuing theory. Thus we find that modeling and analysis of a virtualized DCN using stochastic models are still a preliminary endeavor. This motivates us to model and analyze a virtualized DCell-based DCN using SRN.

We summarize the main contributions of our work as follows:

(i) Modeled a DCell-based DCN for business continuity under two configurations, respectively, consisting of two and three virtualized servers in a $DCell_0$ in a complete manner using SRN.

(ii) Incorporated failure modes and recovery behaviors of hosts, switches, and VMs along with VM live migration within and between $DCell_0$s for the sake of fault tolerance.

(iii) Captured the featured dependencies between components in the system architecture in detail: (a) between hosts and VMs and (b) between switches and VMs.

(iv) Performed detailed analyses of the constructed SRN models in terms of steady state availability, downtime cost, modeling complexity, and sensitivity with respect to major parameters.

Through modeling and analysis, we have found the following:

(i) A virtualized DCell-based DCN with greater number of hosts in a $DCell_0$ can enhance the level of continuity and availability. The DCN with three hosts in a $DCell_0$ can deliver a level of availability over tier 4 in HA standards for a data center [20].

(ii) In a virtualized DCN based on DCell network topology, the recovery of hardware (hosts) and software (VMs) subsystems contributes a major impact on system availability. As the size of the DCN increases, the recovery of software subsystem exposes a more important role versus that of hardware subsystem.

(iii) A bigger size of the VMs in a DCN causes a declining tendency of system availability. Nevertheless, in a more complicated DCN, the influence of VM image size is mitigated.

(iv) The cross-links between the pairs of hosts in different $DCell_0$s in a DCN and their bandwidth are necessary elements to tolerate switch failures, to mitigate system downtime, to improve system overall availability, and thus to secure system operation for business continuity.

(v) The modeling and analysis of the virtualized DCNs in this paper help guide the system design and management of a DCN:

  (a) A thorough adoption of software fault tolerance is necessary in the design of a DCN.

  (b) The effectiveness and readiness of the repair and maintenance services in a DCN need more attention and improvement.

  (c) The trade-off between system availability and performance and the overall cost of networking [21] in a DCN is an important metric in system design.

The rest of this paper is organized as follows. Related work is presented in Section 2. Section 3 introduces a virtualized DCN. Section 4 presents SRN models for the DCNs. The numerical analysis and discussion are presented in Section 5. Finally, Section 6 concludes the paper.

2. Related Work

Design of data center infrastructure is critically demanding in research and development both from academia and industry to deliver cloud-based online apps and services with highest availability. Server-network architecture within a data center thus plays an important role in enhancing agility and reconfigurability of interconnecting different infrastructure resources. In that context, topologies of a DCN contribute major impact on system performance, availability, and scalability to deliver changeable application demands and service requirements [22]. Current DCNs adopt three-layer network topology in which physical servers are interconnected into a rack by a top of rack (ToR) switch, the ToR switches are

networked through end of rack (EoR) switches, and core switches connect these EoR switches together to the external network providers [15]. The topology however confronts a variety of critical issues as a nature: (i) fault-dependency propagation in which a failure of an upper-level switch causes the complete disconnection and unavailability of a number of dependent switches and servers connected to the failed switches and (ii) significant bandwidths that are required to maintain efficient connectivity. To avoid and eliminate the current issues, a number of network topologies have been proposed for alternatives including (i) tree based topologies, such as fat-tree [3, 23, 24] and Clos Network [25] and (ii) recursive-based topologies, such as DCell [4], FiConn [26], BCube [5], and Hyper-BCube [27]. In the work [15], Liu et al. conducted a detailed comparison between the DCNs and concluded that no single topology outperforms the others in all aspects and there will always be trade-offs among cost, performance, and reliability. Among the network topologies, the DCell comes out as a candidate to satisfy the requirements of robustness and connectivity [6], fault tolerance, and scalability in a data center even though the aggregated bottleneck throughput is comparatively low [15]. In this paper, we focus on the DCell-based network topology in consideration of fault tolerance and network availability, which were not considered in the previous work [15]. A DCell [4] is recursively constructed based on the most basic element DCell_0 as follows:

(i) A DCell_0 consists of n physical servers all connected to n-port switch.

(ii) A DCell_1 is composed of $n+1$ DCell_0s. Each server of a DCell_0 in a DCell_1 has two links. One connects to its switch; the other connects to a corresponding server in another DCell_0, complying with a predetermined DCell routing algorithm. Consequently, every pair of DCell_0s in a DCell_1 has exactly a unique link between each other.

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

In this paper, we study two DCell-based DCNs at the level of DCell_1 which, respectively, consist of two and three servers in a DCell_0. We will show through availability as the measure of interest that such DCell-based DCNs expose better ability to tolerate node and switch failures.

High availability (HA) and business continuity (BC) are the key factors in designing enterprise computing systems for a cloud-based business to be successful [28]. Nevertheless, as computing systems with high level of complexity and dependency have been coming out such as Infrastructure-as-a-Service (IaaS), software defined data center (SDDC), and software defined network (SDN), the systems are likely prone to a variety of failures. Severely, a failure of a component may cause a cascading failure or unavailability of a group of other components. For instance, the failure of a switch connecting to a number of physical servers causes the unavailability of the set of servers at the same time. To achieve predetermined levels of HA and BC, which are indicated in service level agreement (SLA) [29, 30] between customers and system owner, the system design has to tolerate any single point of failure in both hardware and software subsystems. The ANSI/TIA-942 [20] presents four tiers; each specifies basic requirements of availability and downtime minutes per year as follows: (i) tier 1 (basic): 99.671% for availability and 1729.224 downtime minutes in a year; (ii) tier 2 (redundant components): 99.74% for availability and 1361.304 downtime minutes in a year; (iii) tier 3 (concurrently maintainable): 99.982% for availability and 94.608 downtime minutes in a year; and (iv) tier 4 (fault tolerant): 99.995% for availability (four nines) and 26.28 downtime minutes in a year. In order to achieve the above levels of HA standards, one may need to adopt server virtualization [11, 31–34] on nodes and apply VM live migration [12, 35, 36] as the means of fault tolerance for nodes and switches. Server virtualization creates and fosters a plurality of VMs on each physical server. With VM live migration mechanism, a VM is not only able to be migrated from a failed host to another host in the same DCell_0 as soon as a host's failure occurs but also it can be migrated from a running host in a DCell_0 to another host in other DCell_0s if the switch in the former DCell_0 fails. In this paper, we will show a comprehensive modeling and analysis of a virtualized DCell-based DCN for high availability and continuity. The analysis results shown in Section 5 reflect that a DCN complying with DCell network topology along with server virtualization and VM live migration mechanism can achieve HA whereas the DCN with a standalone DCell_0 cannot. Furthermore, the overall system availability of the DCell-based DCN surpasses tier 4 (which is a high available and fault tolerant system) in the ANSI/TIA-942 HA standard for a data center.

Sensitivity analysis [10, 18, 37, 38] is used popularly to provide a selection basis and help design system parameters by observing system characteristics and responses with respect to predetermined valuables in order to identify the most impacting factors as well as to detect bottlenecks in system availability. One may adopt two types of sensitivity analysis: (i) nonparametric sensitivity analysis [39], which studies the system responses upon the component addition/removal or modifications of system model and (ii) parametric sensitivity analysis [40], which observes the system behaviors with respect to the variations of given input parameters. The parametric sensitivity analysis has been adopted to assess system performance and reliability/availability upon the effect of changes of given parameters in different systems. Nguyen et al. [10] presented a comprehensive sensitivity analysis of system steady state availability for a virtualized servers system. The thorough study of availability sensitivity with respect to the intervals of software rejuvenation on VMMs and VMs provides a design basis on how to improve availability of a virtualized system in a wiser manner by combining both software rejuvenations at VM and VMM. In the work [18], Matos Jr. et al. applied parametric sensitivity analysis on a redundant computer network system with respect to MTTF and MTTR of every network component to figure out the important and influent factors of network availability. In another work [38], Matos et al. implemented four different sensitivity analysis techniques to identify the parameters with greatest impact on the availability of a mobile cloud computing system. Accordingly, a sensitivity analysis

can be conducted to assess the importance of parameters in the following approaches:

(i) Repeatedly vary one selected parameter at a time while the others are kept constant and observe the system behaviors on the measures of interest with respect to the varying parameter. This approach studies the system responses upon a broad value range of the parameters in consideration.

(ii) For differential sensitivity analysis, compute partial derivatives of the measure of interest with respect to each system parameter. This approach is useful in the case that input values of parameters are assigned in a continuous domain. The differential sensitivity of the system availability A with respect to variable is defined as in (1) or (2) for a scaled sensitivity:

$$S_\tau(A) = \frac{\partial A}{\partial \tau}, \tag{1}$$

$$SS_\tau(A) = \frac{\partial A}{\partial \tau}\left(\frac{\tau}{A}\right). \tag{2}$$

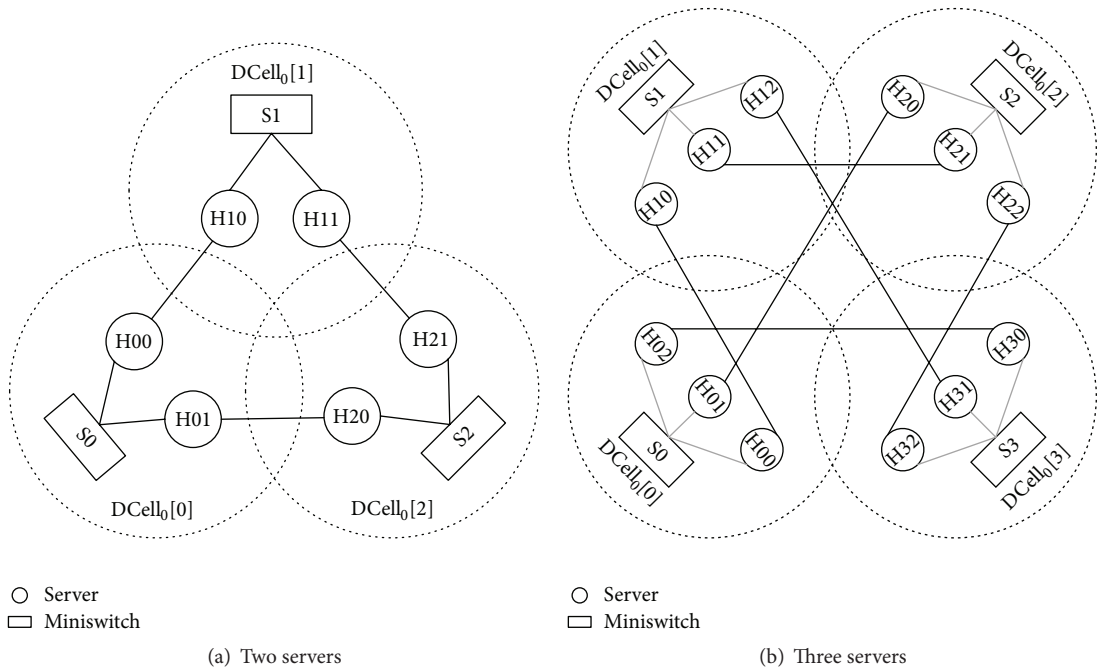
(iii) Calculate the percentage difference in the variation of a parameter from its minimum to maximum values. This technique is designed for integer-valued parameters which are not properly evaluated by the differential sensitivity analysis approach.

(iv) For design of experiments (DOE) [41], simultaneously examine individual and interactive effects of factors on the output measures.

In this paper, we adopt approaches (i) and (ii) to study system behaviors and responses with respect to parameters at the default values and in a broad value domain. The analyses (i) help find the major impact factors on system availability and (ii) study system characteristics upon the variations of parameters and thus (iv) help design system parameters and (v) guide to adopt proactively different tolerance techniques to achieve optimized overall system availability.

There are a few works on sensitivity modeling and analysis of availability for DCNs. Matos Jr. et al. in the work [18] modeled and analyzed the availability of a small-scale computer network using continuous time Markov chain (CTMC). This very first work studied the impact of the failures of network devices (switches and routers) and network links on system availability. The study took into account the redundancy of either network devices or links as a measure to tolerate the aforementioned failures and to improve system availability. The contributions of this work suggest the approach of adopting stochastic models to analyze a DCN. However, as the system scale increases, the CTMC models (where each state in the model is the combination of all states of the components in consideration) likely confront largeness problem or state-space explosion as well as intractable presentation of the model. Furthermore, unplanned redundancy of physical devices is a costly solution especially for DCNs as the number of machines increases vastly. It is needed to organize the physical components in a well-designed network topology for either fault tolerance or high availability and/or performance. Alshahrani and Peyravi in a very first work [19] on modeling and analysis of DCNs attempted to adopt queuing theory to model a typical topology of DCN, fat-tree [3]. This work proposed a detailed analytical model to assess performance metrics of interest (e.g., throughput and delay) of a fat-tree based DCN. The work nevertheless did not take into consideration any type of failures. The system architecture is composed of only network devices for simplification of theoritical formulation. This preliminary work raises a need to conduct further studies on various attributes (e.g., availability, reliability, and performability [42]) of dependability of different DCN topologies. Several other works studied various essential issues of contemparory DCNs. Liu et al. [1, 15] studied fault tolerance characteristics of renowned network topologies of DCNs. Among different topologies of DCNs presented in the works [3–5, 23, 25, 27, 43], DCell topology is pointed out as a typical topology with high scalability and fault tolerance capability [1, 4, 15] with more and more interest in practice. Several works [44–46] presented large-scale emperical studies of failures in typical data centers. The works have charaterized a variety of failures in DCNs such as failures of servers (e.g., hard disk, memory and raid controller failures) and failures of network devices (top-of-rack switch, aggregation switch, and router failures). Some other works [7, 9, 11, 47–50] showed the adoption of renowned virtualization technology and VM migration in computing systems is of paramount importance to achieve high availability and to tolerate unexpected risks or failures.

Based on the above literature review, we find that the modeling and analysis of a DCN are still in initial steps. Previous work attempted to model and analyze DCNs using either stochastic models or queuing theory without an adequate consideration of different system failures and corresponding fault tolerant techniques. Moreover, the system architectures did not incorporate contemporary virtualization technology and VM migration techniques for high availability and effecient fault tolerance in virtualized environment. Our focus in this paper is on the ability of DCell-based DCNs to tolerate any type of risks in the system in order to ensure the system's safety in terms of system operation and availability. We are more interested in risks and measures to tolerate risks for DCNs to achieve high availability (which are critically demanding in design of network in data centers) rather than other attibutes of DCN's dependability. Therefore we choose to study DCell network topology for DCNs which allows us to enhance the system capability of fault tolerance. We attempted to use SRN with a variety of modeling funtionalities in order to capture the dependency between upper and lower level components in the system architecture (for instance, between a physical server and its hosted VMs or between a switch and its connected servers). Furthermore, the SRN models are tractable (literally compared to CTMC) and thus enable us to incorporate various behaviors (failures, VM migration, and interaction between submodels). We will be using SRN to model typical DCell-based DCNs in the next sections.



(a) Two servers (b) Three servers

Figure 1: DCN system architectures.

3. A Virtualized Data Center Network

3.1. System Architectures. The system architectures of a DCell-based DCN are depicted in Figure 1. Figure 1(a) shows the architecture of a DCell-based DCN with two servers in a $DCell_0$ (called DCN2 from now on) whereas Figure 1(b) depicts the architecture of the DCell-based DCN with three servers in a $DCell_0$ (called DCN3). Both DCN2 and DCN3 comply with DCell configuration and routing [4]. Accordingly, DCN2 consists of three $DCell_0$s: $DCell_0[0]$, $DCell_0[1]$, and $DCell_0[2]$. Each of $DCell_0$s comprises two servers (also called host) and a gigabit switch to help clients use the servers. All hosts are virtualized to run a number of virtual machines (VMs). Within a $DCell_0$, gigabit links for high speed data transactions connect the hosts and the corresponding switch. For easy understanding of the SRN models of the system (to be presented in the next sections), we apply a naming convention in which characters represented for a component accompany specific numbers. In particular, the $DCell_0[0]$ consists of the switch S0, the hosts H00 and H01, and their respective virtual machines VM00 and VM01. The naming convention is applied in the same way for $DCell_0[1]$ which comprises S1, {H10, H11}, and {VM10, VM11} and for $DCell_0[2]$ which is also composed of S2, {H20, H21}, and {VM20, VM21}. In the same way of the above descriptions, DCN3 is composed of four $DCell_0$s from $DCell_0[0]$ to $DCell_0[3]$. Each $DCell_0$ in DCN3 consists of one switch and three hosts and each host in turn runs a number of VMs. The notation of components in DCN3 complies with the naming convention aforementioned in DCN2. The network routing is formed between hosts among different $DCell_0$s. Particularly, internal network links in DCN2 are formed between the following pairs of hosts: {H00, H10}; {H01, H20}; and {H11, H21}. Whereas, in DCN3 the links are formed between the pairs: {H00, H10}; {H01, H20}; {H02, H30}; {H11, H21}; {H12, H31}; and {H22, H32}, which comply with DCell routing tactics [4]. We will use these architectures to model and analyze the system availability in the next sections.

3.2. System Behaviors and Assumptions

(i) Operational States. A host and a switch can fail and recover upon the states of their hardware components as similar as the blade server described in [51]. And a VM may go through a variety of complicated states as in [10, 44, 45]. But capturing the different operational states of hosts, VMs, and switches in modeling in a complete manner is beyond our focus and could lead to largeness problem [52] of the models. Hence, the two-states model (up and down states) is used to represent the basic operational states of system subsystems.

(ii) VM Live Migration. The VM live migration technique is employed to tolerate unexpected failures of hosts and switches. In a $DCell_0$, if a host fails, all VMs running on the failed host are immediately migrated onto the remaining hosts with good consideration of load balancing. Moreover, if a switch fails, all VMs operating on the hosts in the $DCell_0$ of that failed switch are instantly migrated to the other hosts of all other remaining $DCell_0$s. For instance, in the case of the DCN2, if the host H00 fails, the VMs running on H00 are live-migrated onto the host H01. When the switch S0 goes down, the live migration processes are triggered instantly to migrate the VMs running on H00 and H01, respectively, onto the host H10 of the $DCell_0[1]$ and onto the host H20 of the $DCell_0[2]$. The descriptions can be applied in the same way for other $DCell_0$s in DCN2 and DCN3. The above VM live

migration mechanisms are used to prevent the VMs from unexpected downtime due to failures of hosts and switches; thus system availability is improved and business continuity is endured. In order to reduce the complexity of system models, it is necessary to assume that the VM live migration processes do not confront any unexpected failures such as data loss and memory errors, during the migration period as captured in some work [35, 53].

(iii) Virtual Machine Monitor (VMM). Hypervisor or VMM is in charge of creating and maintaining virtualization environment to operate the upper VMs. Thus, the operational states of VMs are dependent on the operations of the underlying VMM. The detailed dependencies of a host, a VMM, and a VM are captured in a number of works [10, 44, 45]. Nevertheless, we do not take into account the VMM in modeling for simplification and our focus is on the states of VMs, since user's applications and services run on VMs. We consider the up and down states of VMM as one part of the host's up and down states.

(iv) High Availability. Our systems are designed to deliver HA services to the user. In HA system, the availability is assigned to the cases that more components are in up states. Thus, we assume that repair and maintenance services in data centers are good enough to recover the failed hardware components in advance of the remaining component failures. Particularly in DCN2, we assume the following:

(i) If a host fails in a DCell_0, the remaining host only fails after the recovery of the aforementioned host.

(ii) If the number of failed hosts in the DCN2 is larger or equal to the haft of the total number of hosts, the remaining hosts can fail after one of the failed hosts is recovered.

(iii) If two switches fail, the repair person is summoned and repairs subsequently the failed switch before the failure of the remaining switch.

(iv) A switch and a host are in operation and can fail if there is at least a VM running in the host of the same DCell_0.

In the case of DCN3, we assume the following:

(i) If two hosts fail in a DCell_0, the remaining can fail as either one or both of the failed hosts are recovered.

(ii) If the number of failed hosts in DCN3 is larger or equal to the haft of the total number of hosts, the remaining hosts can fail as one of the failed hosts is recovered.

(iii) If three out of four switches fail, the remaining switch can fail after the recovery of one of the failed switches.

(iv) A switch and a host are in operation and can fail if there is at least a VM running in the host of the same DCell_0. The purposes of these assumptions are to reduce the largeness of the model and to mitigate the cases with very low probability to occur in HA system.

(v) Distributions. The time to occurrence of any event in actual computing system may follow different types of probability distribution [54]. However, we can make appropriate distribution assumptions for every transition so that the analytical system model is closer to the practical system. In this paper, we choose to use exponential distribution for simplification in modeling and analysis as a common option in a large number of papers [10, 44, 45].

The above system behaviors and assumptions are all taken into consideration in the modeling of the DCNs to be carried out and described in detail in Section 4.

4. Stochastic Reward Net Models

4.1. System Models. The SRN system models of DCN2 and DCN3 are, respectively, depicted in Figures 2 and 3. The system models are composed of partial models including host models, switch models, and VM subsystem model as chronologically named from Figure 2(a to g) in SRN system model for DCN2 and from Figure 3(a to q) in SRN system model for DCN3. In consideration of system availability measures, we use two-states model (up and down states) to model hosts, switches, and VMs for the sake of modeling simplification. Our sharp focus in modeling is on the dependency between components and fault tolerant behaviors for high availability in the case of any component's failure. We will describe the model of a specific component as an example to refer to the other similar components. Then the model integration and dependency will be presented subsequently. The transitions of tokens within the models are conducted stochastically by enabling/disabling the timed/immediate transitions based on the predefined behaviors. The combinations of all tokens' locations in the models represent the system's respective states. We list down all the states of every component and possible locations of tokens in the models as in Table 1. In order to capture exact predefined system behaviors we apply a set of guard functions [55–58] attached to every transitions in the models to control the transitions of tokens.

4.2. SRN Models of Hosts, VMs, and Switches. Figure 4 shows the two-state SRN models of selected host, VM, and switch. In the assumption, we mentioned that the characteristics and configurations of all hosts, VMs, and switches are assumed to be identical initially. Thus, we can use the two-state SRN models to capture up and down states of the component in regard of availability measures. We describe the modeling of the host H00, the VM00, and the switch S0 as the examples to refer to the modeling of the other identical hosts, VMs, and switches in both SRN system models of DCN2 and DCN3.

Figure 4(a) depicts the modeling of a host with repair actions. Initially, a host is considered in running state depicted by a token in up state P_{H00up}. A virtualized host in DCN may undergo an expected failure or maintenance period after a specific time with MTTF $1/\lambda_{\text{H}}$. In this case, the transition T_{H00f} is triggered to fire and the token in P_{H00up} is removed and deposited in P_{H00dn}. As the host goes down, a repair person is summoned to recover the host. After the repair, the transition T_{H00r} is enabled and the token in

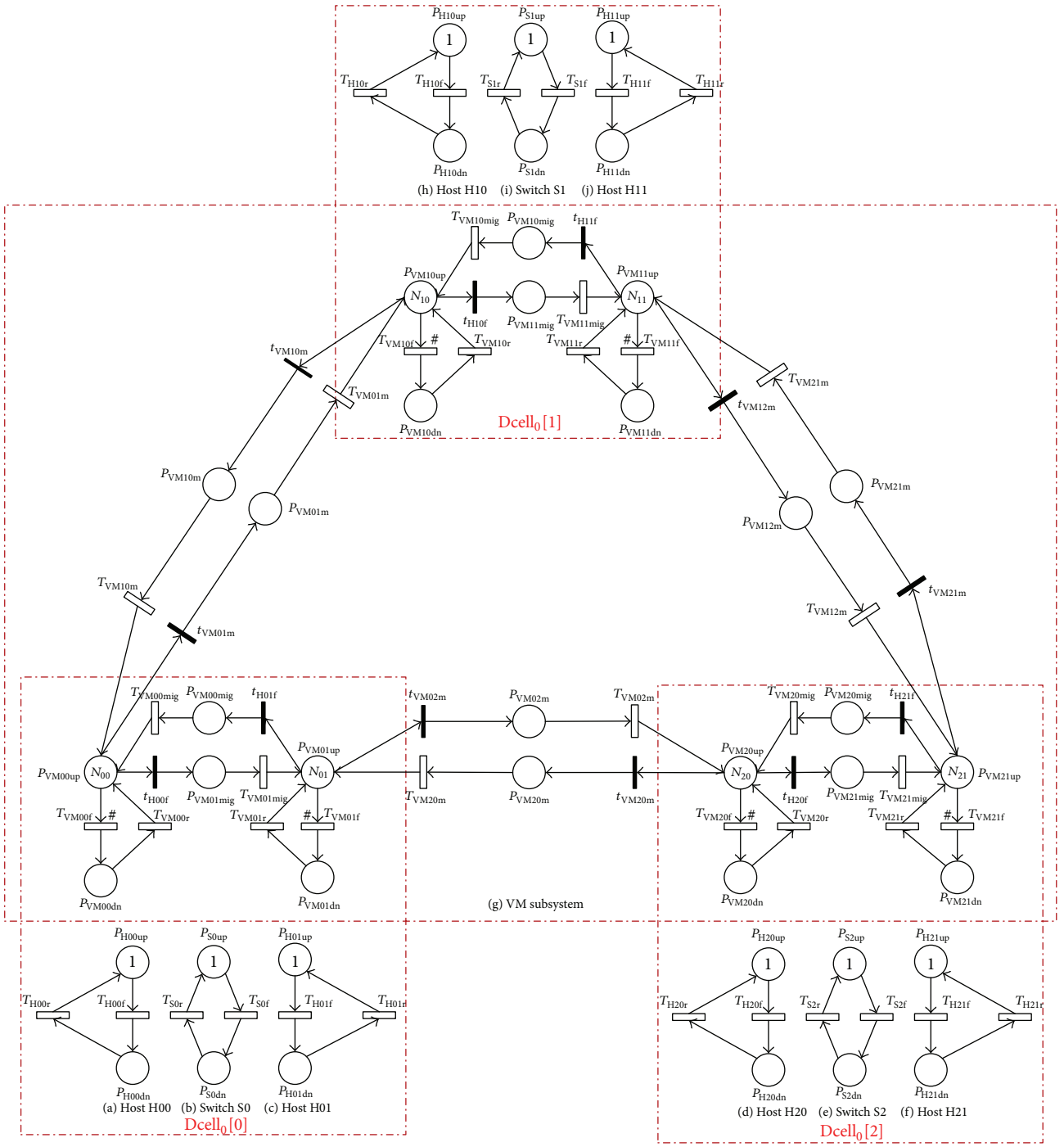


Figure 2: SRN model of a DCN with two servers in a DCell_0.

P_{H00dn} is removed and deposited in P_{H00up}. The host returns operational.

Figure 4(b) captures the behaviors of VMs running on host H00. Assume that there are initially N_{00} in running states. As time goes by, a VM can fail with a failure rate λ_{VM}. The transition is fired subsequently and one token in P_{VM00up} is removed and deposited in P_{VM00dn}. The VM goes down. Because of the competition between the VMs in up state to fail, the failure rate of the running VMs at a time depends on the number of VMs or, in other word, the number of tokens in the place P_{VM00up}. Therefore, we apply marking dependence on the transition T_{VM00f} represented by the marker "#." The VMs in downstate are repaired in sequence by software or by a repair person. The repaired VM restarts to healthy state. This repair action is captured by firing the transition T_{VM00r}; then a token in P_{VM00dn} is taken out and deposited in P_{VM00up}.

Figure 4(c) presents the failure and repair action of a switch in modeling. At the beginning the switch is considered

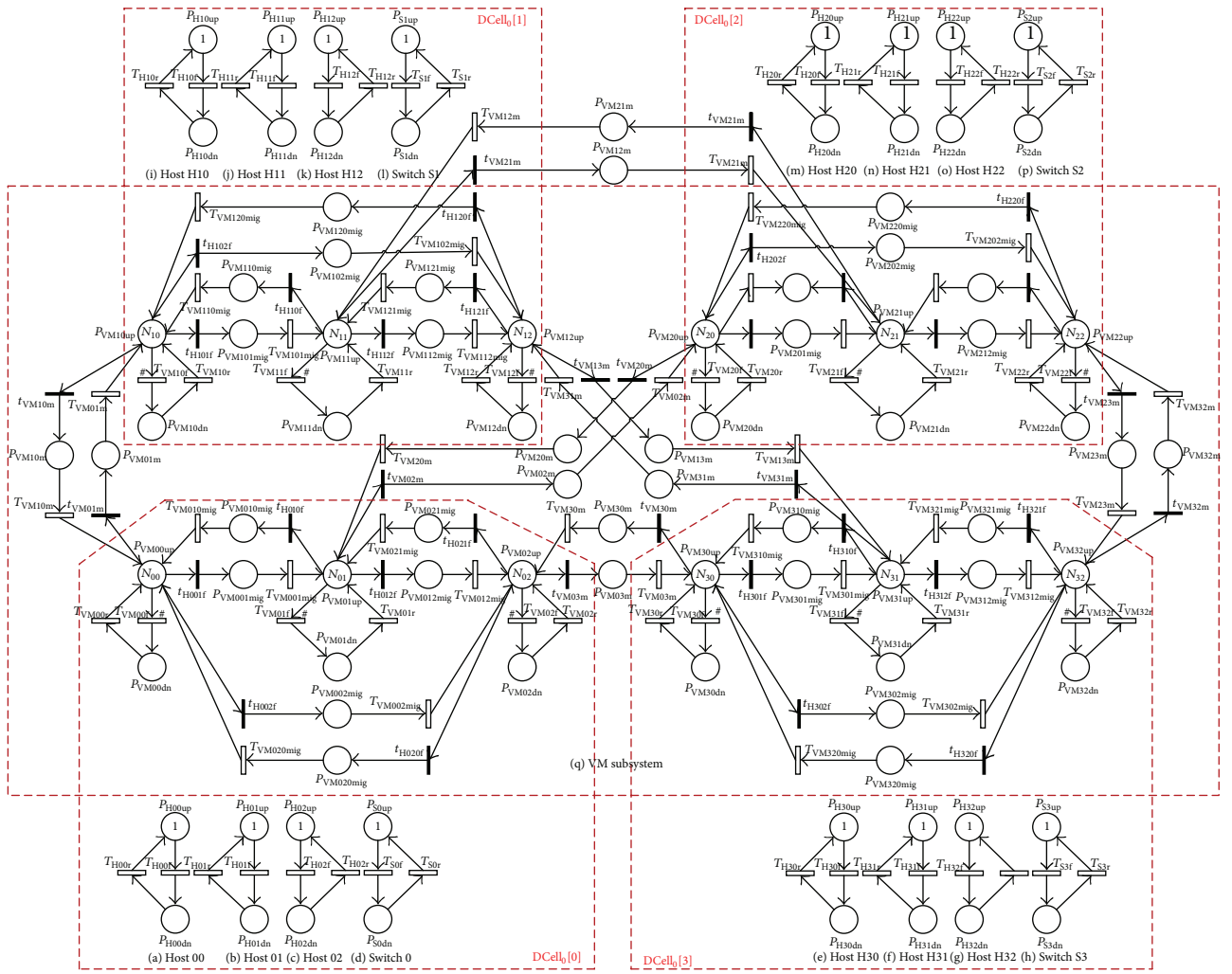


Figure 3: SRN model of a DCN with three servers in a $DCell_0$.

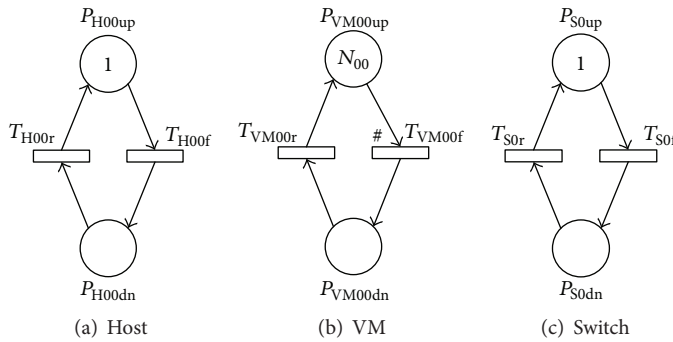


Figure 4: SRN models of hosts, VMs, and switches.

in healthy state depicted by a token in P_{S0up}. After some time, it may fail, the transition T_{S0f} is triggered to fire, and the token in P_{S0up} is taken out and deposited in P_{S0dn}. The switch fails consequently. After repairing the failed switch, the transition T_{S0r} is enabled and the token in P_{S0dn} is removed and deposited in P_{S0up}. The switch starts running normally.

4.3. SRN Models of a Standalone $DCell_0$. Figures 5 and 6, respectively, depict the SRN models of the $DCell_0$s comprising two and three hosts (hereinafter called DCN0 and DCN1), which are the basic units to construct the DCell-based DCN2 and DCN3. The DCN0 and DCN1 are actually the $DCell_0[0]$ taken out for an example of modeling

Table 1: Description of places in SRN system models of DCN2 and DCN3.

Place	Description
P_{H00up}, P_{H01up}, P_{H02up}, P_{H10up}, P_{H11up}, P_{H12up}, P_{H20up}, P_{H21up}, P_{H22up}, P_{H30up}, P_{H31up}, and P_{H32up}	Running states of respective hosts: H00, H01, H02, H10, H11, H12, H20, H21, H22, H30, H31, and H32.
P_{H00dn}, P_{H01dn}, P_{H02dn}, P_{H10dn}, P_{H11dn}, P_{H12dn}, P_{H20dn}, P_{H21dn}, P_{H22dn}, P_{H30dn}, P_{H31dn}, and P_{H32dn}	Down states of respective hosts: H00, H01, H02, H10, H11, H12, H20, H21, H22, H30, H31, and H32.
P_{S0up}, P_{S1up}, P_{S2up}, and P_{S3up}	Running states of respective switches: S0, S1, S2, and S3.
P_{S0dn}, P_{S1dn}, P_{S2dn}, and P_{S3dn}	Down states of respective switches: S0, S1, S2, and S3.
P_{VM00up}, P_{VM01up}, P_{VM02up}, P_{VM10up}, P_{VM11up}, P_{VM12up}, P_{VM20up}, P_{VM21up}, P_{VM22up}, P_{VM30up}, P_{VM31up}, and P_{VM32up}	Running states of respective VMs: VM00, VM01, VM02, VM10, VM11, VM12, VM20, VM21, VM22, VM30, VM31, and VM32.
P_{VM00dn}, P_{VM01dn}, P_{VM02dn}, P_{VM10dn}, P_{VM11dn}, P_{VM12dn}, P_{VM20dn}, P_{VM21dn}, P_{VM22dn}, P_{VM30dn}, P_{VM31dn}, and P_{VM32dn}	Down states of respective VMs: VM00, VM01, VM02, VM10, VM11, VM12, VM20, VM21, VM22, VM30, VM31, and VM32.
$P_{VM00mig}$ and $P_{VM01mig}$; $P_{VM10mig}$ and $P_{VM11mig}$; $P_{VM20mig}$ and $P_{VM21mig}$	Intermediate states of VM migration processes in $DCell_0[0]$, $DCell_0[1]$, and $DCell_0[2]$ in DCN2, respectively, from H01 to H00 and from H00 to H01, from H11 to H10 and from H10 to H11, and from H21 to H20 and from H20 to H21.
$P_{VM001mig}$ and $P_{VM010mig}$; $P_{VM012mig}$ and $P_{VM021mig}$; $P_{VM002mig}$ and $P_{VM020mig}$	Intermediates states of VM migration processes in $DCell_0[0]$ in DCN3, respectively, from H00 to H01 and from H10 to H00, from H01 to H02 and from H02 to H01, and from H00 to H02 and from H02 to H00
$P_{VM101mig}$ and $P_{VM110mig}$; $P_{VM112mig}$ and $P_{VM121mig}$; $P_{VM102mig}$ and $P_{VM120mig}$	Intermediate states of VM migration processes in $DCell_0[1]$ in DCN3, respectively, from H10 to H11 and from H11 to H10, from H11 to H12 and from H12 to H11, and from H10 to H12 and from H12 to H10.
$P_{VM201mig}$ and $P_{VM210mig}$; $P_{VM212mig}$ and $P_{VM221mig}$; $P_{VM202mig}$ and $P_{VM220mig}$	Intermediate states of VM migration processes in $DCell_0[2]$ in DCN3, respectively, from H20 to H21 and from H21 to H20, from H21 to H22 and from H22 to H21, and from H20 to H22 and from H22 to H20.
$P_{VM301mig}$ and $P_{VM310mig}$; $P_{VM312mig}$ and $P_{VM321mig}$; $P_{VM302mig}$ and $P_{VM320mig}$	Intermediate states of VM migration processes in $DCell_0[3]$ in DCN3, respectively, from H30 to H31 and from H31 to H30, from H31 to H32 and from H32 to H31, and from H30 to H32 and from H32 to H30.
P_{VM01m} and P_{VM10m}	Intermediate states of VM migration processes between $DCell_0[0]$ and $DCell_0[1]$ in DCN2 and DCN3, respectively, from H00 to H10 and from H10 to H00.
P_{VM12m} and P_{VM21m}	Intermediate states of VM migration processes between $DCell_0[1]$ and $DCell_0[2]$ in DCN2 and DCN3, respectively, from H11 to H21 and from H21 to H11.
P_{VM20m} and P_{VM02m}	Intermediate states of VM migration processes between $DCell_0[2]$ and $DCell_0[0]$ in DCN2 and DCN3, respectively, from H20 to H01 and from H01 to H20.
P_{VM03m} and P_{VM30m}	Intermediate states of VM migration processes between $DCell_0[0]$ and $DCell_0[3]$ in DCN3, respectively, from H02 to H30 and from H30 to H02.
P_{VM13m} and P_{VM31m}	Intermediate states of VM migration processes between $DCell_0[1]$ and $DCell_0[3]$ in DCN3, respectively, from H12 to H31 and from H31 to H12.
P_{VM23m} and P_{VM32m}	Intermediate states of VM migration processes between $DCell_0[2]$ and $DCell_0[3]$ in DCN3, respectively, from H22 to H32 and from H32 to H22.

description, respectively, in the DCN2 and DCN3. The SRN model of DCN0 in Figure 5 consists of host models of the hosts H00 (Figure 5(a)) and H01 (Figure 5(c)), switch model of the switch S0 (Figure 5(b)), and VM models of the VM00 and VM01 (Figure 5(d)). The modeling of these partial components can be referred to the description of the corresponding models in Figure 4. Henceforth we describe the dependency of the VM model upon the host and switch models. In particular, we apply VM live migration as a fault tolerant technique to avoid the downtime of VMs because of their host's failures. Initially, all components are in up state depicted by the tokens in P_{H00up}, P_{H01up}, P_{S0up},

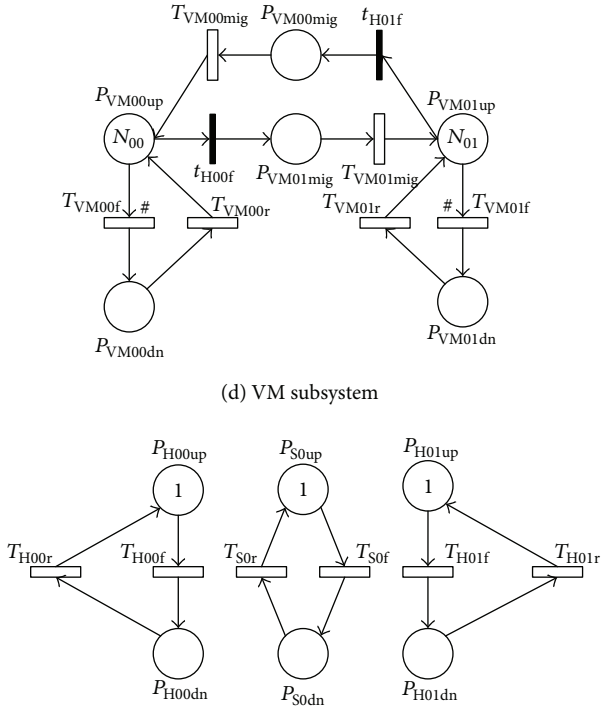


FIGURE 5: SRN models of a $DCell_0$ with two servers.

P_{VM00up}, and P_{VM01up}. At a certain time, the host H00 may fail, which is represented by a token in P_{H00dn}. The failure and repair transitions T_{VM00f} and T_{VM00r} are disabled. All the VMs running on the host H00 consequently are triggered to undergo a live migration process. This behavior is captured by enabling the immediate transition t_{H00f}. The tokens in P_{VM00up} are taken out and deposited in the intermediate place $P_{VM01mig}$. At this point, the VM migration processes start by enabling the transition $T_{VM01mig}$. A token in the place $P_{VM01mig}$ is removed and deposited in the place P_{VM01up} one after another. Thus, the VMs in the failed host H00 are all live-migrated onto the operational host H01 in the same $DCell_0$. In the case that the host H01 fails in the progress of VM migration, the migration processes are interrupted and halted until the failed host H01 is recovered completely. The VMs' image files and related data are stored in the $DCell_0$'s storage, represented by the tokens in the intermediate place $P_{VM01mig}$. Hence, the transition $T_{VM01mig}$ is disabled until the completion of the failed host H01's recovery processes. Based on the above description, we can refer to the case of the host H01's failure. As soon as the host H01 fails (represented by a token in P_{H01dn}) but the host H00 still runs (one token in P_{H00up}), the VMs running on the host H01 are live-migrated onto the host H00 with the same aforementioned processes. The immediate transition t_{H01f} is triggered to fire. All the tokens in P_{VM01up} are taken out and deposited in $P_{VM10mig}$. The migration process of VMs is carried out in sequence as long as the host H00 is operational. If the host H00 fails during the migration of VMs from the failed host H01, the transition $T_{VM10mig}$ is disabled and the VM migration processes stop until the host H00 is recovered. Furthermore, if both hosts H00 and H01 go down along with each other, the running VMs' image file and related data are stored on a shared memory, which is captured by a number of VMs in the places $P_{VM01mig}$ and $P_{VM10mig}$ previously taken out from the respective places P_{VM00up} and P_{VM01up}. Then all the transitions in the VM subsystem model (Figure 5(d)) are disabled to stop completely the VMs' operations. In addition, if the switch S0 fails (a token resides in P_{S0dn}), the running VMs on the hosts H00 and H01 are live-migrated to the respective hosts in the other $DCell_0$s upon the network routing presented in the system architecture in Figure 1(a). This behavior is presented in detail in the next section. Modeling description of the DCN1 model in Figure 6 is carried out in detail as the above description of the DCN0 model in Figure 5 with good consideration on the notation alteration. The DCN1 model consists of the host models of the hosts H00, H01, and H02; the switch model of the switch S0 and the VM models of the VM00, VM01, and VM02, respectively, hosted on the aforementioned hosts. The dependency and behaviors of VM subsystem upon the operational states of the hosts and switch are similar as described in the DCN0 model. The VM live migration processes are conducted between the two among three hosts. If a host fails, the running VMs on the failed host are live-migrated to the two remaining hosts in consideration of balancing the number of VMs on each host. If a switch goes down, the running VMs on each host are live-migrated to the corresponding hosts in the other $DCell_0$s through the cross-links between $DCell_0$s according to the network routing showed in the DCN3 system architecture in Figure 1(b).

4.4. System Model Integration. The models of DCN2 in Figure 2 and of DCN3 in Figure 3 are made of, respectively, three DCN0s and four DCN1s complying with the DCell-based network routing topologies as in the system architecture in Figure 1. The modeling descriptions of every component and $DCell_0$ units are carried out based on the detailed descriptions of partial component models in Figures 4, 5, and 6. In this section, we show the features of DCell-based DCN upon system model integration. In consideration of a standalone $DCell_0$, if its switch undergoes a downtime period because of unexpected failure or planned maintenance, the communication between computing machines in the $DCell_0$ and system users is disconnected as a result. To avoid this adverse situation, in DCN2 and DCN3 the computing VMs are live-migrated to other $DCell_0$s through the cross-links between hosts from different $DCell_0$. In particular, in the DCN2, the $DCell_0[0]$ connects to the $DCell_0[1]$ via the link between the hosts H00 and H10 and to the $DCell_0[2]$ via the link between the hosts H00 and H20. In turn, the $DCell_0[1]$ connects to the $DCell_0[2]$ via the link between the hosts H11 and H21. In the DCN3, the above description goes in similar way in which a $DCell_0$ connects to three remaining $DCell_0$s via different links between the pairs of specific hosts. We take the failure of the switch S0 in the $DCell_0[0]$-DCN2 as an example to describe the system behaviors and interactions between $DCell_0$s upon the failure of switches. As the switch S0 fails depicted by a token in the place P_{S0dn} in Figure 2, all

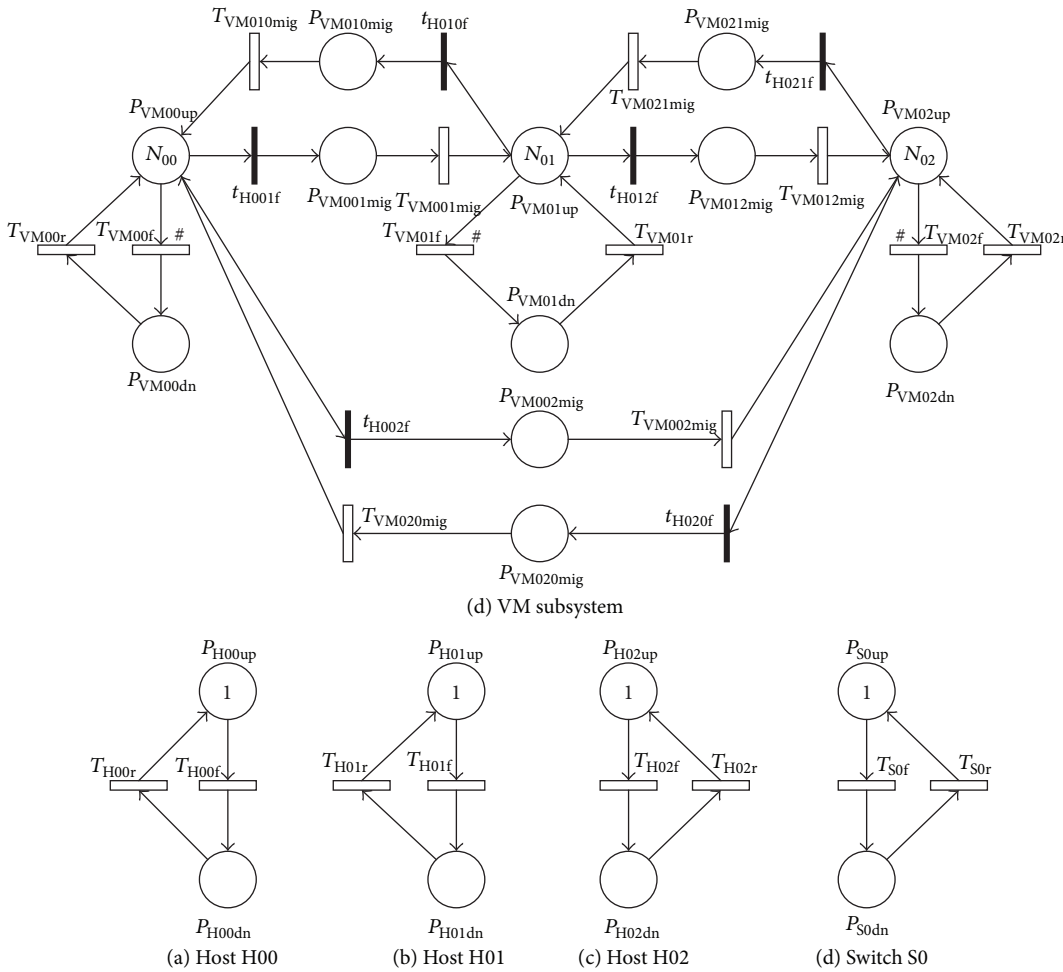


FIGURE 6: SRN models of a $DCell_0$ with three servers.

the running VMs on the hosts H00 and H01 (represented by the tokens residing in the places P_{VM00up} and P_{VM01up}) are live-migrated, respectively, to the hosts H10 in $DCell_0[1]$ and H20 in $DCell_0[2]$. To capture these behaviors in modeling, the immediate transitions t_{VM01m} and t_{VM02m} are triggered to fire as soon as the token in P_{S0up} is removed and deposited in P_{S0dn}. Subsequently, the tokens in the places P_{VM00up} and P_{VM01up} are taken out and deposited, respectively, in the places P_{VM01m} and P_{VM02m}. At this point, the VMs' image files and related data stored in the local memory system of the $DCell_0[0]$ are organized to be migrated from $DCell_0[0]$ to $DCell_0[1]$ and $DCell_0[2]$. The timed transitions T_{VM01m} and T_{VM02m} are enabled to start migration processes. After the completion of the VM migration processes, the VMs hosted on the host H00 in $DCell_0[0]$ with the failure of the switch S0 now operate on the host H10 in $DCell_0[1]$ and the VMs hosted on the host H01 in $DCell_0[0]$ with the failure of the switch S0 now run on the host H20 in$DCell_0[2]$. Under the same reasoning, we can describe the live migration mechanism of VMs from $DCell_0[1]$ to $DCell_0[0]$ and $DCell_0[2]$ upon the failure of the switch S1 and also from $DCell_0[2]$ to $DCell_0[0]$ and $DCell_0[1]$ upon the failure of the switch S2. In the DCN3 system model, as soon as the switch S0 in $DCell_0[0]$ fails, the tokens running on the hosts H00, H01, and H02 (depicted by the tokens in the places P_{VM00up}, P_{VM01up}, P_{VM02up}) are, respectively, migrated to the hosts H10 in $DCell_0[1]$, H20 in $DCell_0[2]$, and H30 in $DCell_0[3]$ (captured by, resp., depositing the tokens in P_{VM10up}, P_{VM20up}, and P_{VM30up}). Based on the above detailed description, the migration of VMs from the other $DCell_0$s upon switch failures can be conducted accordingly.

5. Numerical Results

The SRN models of the DCNs are implemented in Stochastic Petri Net Package (SPNP) [57]. SPNP provides two ways to implement the SRN models: (i) raw input language for SPNP called CSPL (C-based SPN Language), an extension of the C programming language with a variety of application programming interfaces (API) for easier description of SRN models; and (ii) a Graphical User Interface (GUI) for intuitive specification of the SRN models, which later is converted into CSPL automatically by the software itself. The models are converted into Markov Reward Model (MRM) and then solved by using analytic-numeric methods with regard to specific metrics of interest. We use GUI to construct and

verify the correctness of the SRN models and CSPL input source to solve the models and generate various numerical analysis results as well as to investigate the complexity of those analyses. Our metrics of interest for analyses include (i) steady state availability (SSA), (ii) downtime cost, and (iii) sensitivity of SSA with respect to major impacting parameters. Default values of parameters used in modeling are provided in Table 2 based on previous works [10, 50, 58, 59].

To investigate the capability of the DCNs to assure business continuity, we initiate one VM to run on each host of the $\mathrm{DCell}_0[0]$ in either DCN2 or DCN3 at the beginning; and none of VMs is initialized on all the other hosts. In general, the DCell-based DCNs can maintain business operations on the aforementioned VMs even in the case of switch failures by migrating the VMs onto the other hosts of all the other DCell_0s. Thus the overall system availability is improved apparently. These features of the DCell-based DCNs are shown by the numerical analysis results in the next subsections.

5.1. Steady State Analysis. The steady state analyses are carried out along with the downtime and cost analyses for four case studies from (I) to (IV) as in Tables 3 and 4. In order to compute the measures of interest using SPNP, we define the requirements of our systems' availability as follows: (i) there is at least a VM running in a certain DCell_0 and (ii) the switch in the DCell_0 stays in operational state. The requirements are to ensure that there is at least a connection between system users and running computing units. Based on the predetermined requirements, we define the reward functions to compute the system availability for the four DCNs as follows:

$$
\begin{aligned}
A_{\mathrm{DCN0}} &= \begin{cases} 1, & \text{if } \left(\#P_{\mathrm{VM00up}} + \#P_{\mathrm{VM01up}} > 0\right) \&\& \left(\#P_{\mathrm{S0up}} == 1\right), \\ 0, & \text{otherwise,} \end{cases} \\
A_{\mathrm{DCN1}} &= \begin{cases} 1, & \text{if } \left(\#P_{\mathrm{VM00up}} + \#P_{\mathrm{VM01up}} + \#P_{\mathrm{VM02up}} > 0\right) \&\& \left(\#P_{\mathrm{S0up}} == 1\right), \\ 0, & \text{otherwise,} \end{cases} \\
A_{\mathrm{DCN2}} &= \begin{cases} 1, & \text{if } \left\{\left(\#P_{\mathrm{VM00up}} + \#P_{\mathrm{VM01up}} > 0\right) \&\& \left(\#P_{\mathrm{S0up}} == 1\right)\right\} \\ & \quad \| \left\{\left(\#P_{\mathrm{VM10up}} + \#P_{\mathrm{VM11up}} > 0\right) \&\& \left(\#P_{\mathrm{S1up}} == 1\right)\right\} \\ & \quad \| \left\{\left(\#P_{\mathrm{VM20up}} + \#P_{\mathrm{VM21up}} > 0\right) \&\& \left(\#P_{\mathrm{S2up}} == 1\right)\right\}, \\ 0, & \text{otherwise,} \end{cases} \\
A_{\mathrm{DCN3}} &= \begin{cases} 1, & \text{if } \left\{\left(\#P_{\mathrm{VM00up}} + \#P_{\mathrm{VM01up}} + \#P_{\mathrm{VM02up}} > 0\right) \&\& \left(\#P_{\mathrm{S0up}} == 1\right)\right\} \\ & \quad \| \left\{\left(\#P_{\mathrm{VM10up}} + \#P_{\mathrm{VM11up}} + \#P_{\mathrm{VM12up}} > 0\right) \&\& \left(\#P_{\mathrm{S1up}} == 1\right)\right\} \\ & \quad \| \left\{\left(\#P_{\mathrm{VM20up}} + \#P_{\mathrm{VM21up}} + \#P_{\mathrm{VM22up}} > 0\right) \&\& \left(\#P_{\mathrm{S2up}} == 1\right)\right\} \\ & \quad \| \left\{\left(\#P_{\mathrm{VM30up}} + \#P_{\mathrm{VM31up}} + \#P_{\mathrm{VM32up}} > 0\right) \&\& \left(\#P_{\mathrm{S3up}} == 1\right)\right\}, \\ 0, & \text{otherwise.} \end{cases}
\end{aligned} \tag{3}
$$

The numerical results of the steady state analyses and downtime cost analyses with default parameters are shown in Table 5. We assume that a minute of system downtime incurs a penalty of 16,000 USD on the system owner according the SLA signed with customers [60]. The number of nines (a correspondence to availability, nines $= -\log(1 - A)$ [58]) is used to present the improvement and change of steady state availability in an intuitive way. The results show that the adoption of DCell-based architectures improves significantly the system availability and thus decreases vastly the downtime and the corresponding downtime cost. Particularly, the DCN of a DCell0 with two hosts (DCN0 in Figure 5) has the state availability at correspondingly about 2.55 of nines; thus the downtime in a year is at a huge number of 1450.4 minutes and the system owner must bear 23,206,942 USD per year for this system's performance. If we adopt the DCell-based architecture in Figure 3 (DCN3), the system's steady state availability improves vastly with the corresponding number of nines at about 5.19 (almost double compared to DCN0), the system downtime drops off at about 3.4 minutes in a year, and thus the incurred cost now is only 53,959 USD per year. This analysis results reflect the efficiency of the DCell-based DCN in terms of fault tolerance to achieve high availability and mitigate system downtime in comparison with the normal DCN without the adoption of DCell topology. In comparison, the DCNs with three hosts in a DCell_0 (e.g., DCN3 and DCN1) gain relatively higher availability than the respective DCNs with two hosts in a DCell_0 (e.g., DCN2 and DCN0). This is to say that an increase of number of VMs can benefit the system owner to provide higher availability to customers.

Table 2: Parameter default values used in the analyses.

Parameters	Description	Assigned transitions	Mean time/values
λ_H	Host failure rate	T_{H00f}, T_{H01f}, T_{H02f}, T_{H10f}, T_{H11f}, T_{H12f}, T_{H20f}, T_{H21f}, T_{H22f}, T_{H30f}, T_{H31f}, and T_{H32f}	800 hours
μ_H	Host repair rate	T_{H00r}, T_{H01r}, T_{H02r}, T_{H10r}, T_{H11r}, T_{H12r}, T_{H20r}, T_{H21r}, T_{H22r}, T_{H30r}, T_{H31r}, and T_{H32r}	9.8 hours
λ_{VM}	VM failure rate	T_{VM00f}, T_{VM01f}, T_{VM02f}, T_{VM10f}, T_{VM11f}, T_{VM12f}, T_{VM20f}, T_{VM21f}, T_{VM22f}, T_{VM30f}, T_{VM31f}, and T_{VM32f}	4 months
μ_{VM}	VM repair rate	T_{VM00r}, T_{VM01r}, T_{VM02r}, T_{VM10r}, T_{VM11r}, T_{VM12r}, T_{VM20r}, T_{VM21r}, T_{VM22r}, T_{VM30r}, T_{VM31r}, and T_{VM32r}	30 mins
λ_S	Switch failure rate	T_{S0f}, T_{S1f}, and T_{S2f}	1 year
μ_S	Switch repair rate	T_{S0r}, T_{S1r}, and T_{S2r}	24 hours
ω_{mig}	Network bandwidth within a $DCell_0$	$T_{VM00mig}$, $T_{VM01mig}$, $T_{VM10mig}$, $T_{VM11mig}$, $T_{VM20mig}$, $T_{VM21mig}$ $T_{VM001mig}$, $T_{VM010mig}$, $T_{VM002mig}$, $T_{VM020mig}$, $T_{VM012mig}$, $T_{VM021mig}$ $T_{VM101mig}$, $T_{VM110mig}$, $T_{VM102mig}$, $T_{VM120mig}$, $T_{VM112mig}$, $T_{VM121mig}$ $T_{VM201mig}$, $T_{VM210mig}$, $T_{VM202mig}$, $T_{VM220mig}$, $T_{VM212mig}$, $T_{VM221mig}$ $T_{VM301mig}$, $T_{VM310mig}$, $T_{VM302mig}$, $T_{VM320mig}$, $T_{VM312mig}$, and $T_{VM321mig}$	1 Gb/s
ω_m	Network bandwidth between two $DCell_0$s	T_{VM01m}, T_{VM10m}, T_{VM02m}, T_{VM20m}, T_{VM03m}, T_{VM30m}, T_{VM12m}, T_{VM21m}, T_{VM13m}, T_{VM31m}, T_{VM23m}, and T_{VM32m}	256 Mb/s
S_{VM}	Memory size of a VM		10 GB
N_{00}, N_{01}, N_{02}	Number of VMs running on respective hosts H00, H01, and H02		1
N_{10}, N_{11}, N_{12}, N_{20}, N_{21}, N_{22}, N_{30}, N_{31}, N_{32}	Number of VMs running on respective hosts H10, H11, H12, H20, H21, H22, H30, H31, and H32		0

Table 3: Case studies in steady state analyses.

Case	Description
I	A standalone $DCell_0$ with two servers (DCN0)
II	A standalone $DCell_0$ with three servers (DCN1)
III	A DCN with two servers in a $DCell_0$ (DCN2)
IV	A DCN with three servers in a $DCell_0$ (DCN3)

To observe the impact of the number of VMs (n_{VM}) on the steady state availability of DCNs, we conduct the analyses with different values of n_{VM} (from 1 to 6) until the SPNP suffers unexpected memory computation errors (m.e). Table 5 shows the analysis results of steady state availabilities and their corresponding number of nines. In all cases, the increase of n_{VM} slightly gains higher but not significantly system availability for the DCNs.

The adoption of DCell network topology and the increase of number of VMs in DCNs do achieve significantly higher availability for the systems. Nevertheless, it is costly and time-consuming to model and analyze such complicated systems. Table 6 points out the complexity of the analyses using two measures: (i) number of tangible markings and (ii) number of marking-to-marking transitions. As shown clearly, the number of VMs exposes a major influence on the system complexity in modeling and analysis, especially for the systems under the adoption of DCell network topology (DCN2 and DCN3). For DCN0 and DCN1 (without adoption of DCell), the system complexity increases from tens or hundreds to about hundreds or thousands of markings and transitions as the n_{VM} increases from 1 to 6. Whereas in the cases of DCN2 and DCN3, the system complexity boosts up from tens to tens of millions of markings and marking transitions as n_{VM} increases. The vast increase of the system complexity quickly causes memory errors in computation. The DCN2 SRN model suffers unexpected memory errors as the number of marking transitions is at tens of millions. The memory errors in analysis of the DCN3 SRN model occur

Table 4: Steady state and downtime cost analyses.

Case	Type	Steady state availability	Number of nines	Downtime per year (minutes)	Downtime cost per year (USD)
I	DCN0	0.997240422469	2.55	1450.4	23,206,943
II	DCN1	0.997259841407	2.56	1440.2	23,043,637
III	DCN2	0.999950276761	4.30	26.1	418,152
IV	DCN3	0.999993583541	5.19	3.4	53,959

Table 5: Impact of total number of VMs on system steady state availability.

n_{VM}	DCN0		DCN1		DCN2		DCN3	
1	0.997064755072	2.532356	0.997077756809	2.5343	0.999773875854	3.646	0.999803564319	3.71
2	0.997240422469	2.559157	0.997257682983	2.5619	0.999950276761	4.303	0.999989752473	4.99
3	0.997240488479	2.559168	0.997259841407	2.5622	0.999950574780	4.306	0.999993583541	5.19
4	0.997240519634	2.559173	0.997261106490	2.5624	0.999950839446	4.308	m.e	m.e
5	0.997240550678	2.559178	0.997261943932	2.5626	0.999951101800	4.311	m.e	m.e
6	0.997240759564	2.559210	0.997262539658	2.5627	m.e	m.e	m.e	m.e

m.e: memory error.

Table 6: Analysis complexity of tangible markings and marking-to-marking transitions.

Case	Type	1		2		3	
		(i)	(ii)	(i)	(ii)	(i)	(ii)
I	DCN0	26	54	70	180	150	440
II	DCN1	80	198	272	820	664	2270
III	DCN2	91	196	5546	23015	64005	342328
IV	DCN3	365	1050	104567	610066	4224477	32140034
Case	Type	4		5		6	
		(i)	(ii)	(i)	(ii)	(i)	(ii)
I	DCN0	280	900	476	280	900	476
II	DCN1	1408	5248	2710	1408	5248	2710
III	DCN2	403859	2485761	1862923	403859	2485761	1862923
IV	DCN3	m.e		m.e		m.e	

m.e: memory error.

Table 7: Sensitivity of system availability with default parameters.

Parameters	DCN2	DCN3
λ_H	$6.43 \cdot E-08$	$6.12 \cdot E-09$
μ_H	$-2.47 \cdot E-06$	$-2.08 \cdot E-08$
λ_{VM}	$1.66 \cdot E-11$	$2.17 \cdot E-11$
μ_{VM}	$-9.46 \cdot E-08$	$-1.24 \cdot E-07$
λ_S	$1.16 \cdot E-09$	$6.32 \cdot E-10$
μ_S	$-1.30 \cdot E-08$	$-2.54 \cdot E-09$
ω_{mig}	$9.63 \cdot E-09$	$4.55 \cdot E-09$
ω_m	$3.85 \cdot E-08$	$2.15 \cdot E-08$
S_{VM}	$-9.87 \cdot E-07$	$-5.51 \cdot E-07$

as n_{VM} is larger than 3 and thus the complexity could reach hundreds of millions of markings and transitions.

5.2. Sensitivity Analysis. The major purposes of sensitivity analysis in this study are (i) to optimize system design and (ii) to pinpoint the bottlenecks regarding availability, performance, and performability of the systems. Therefore, we conduct a variety of parametric sensitivity analyses of the DCN2 and DCN3 SRN models with respect to the major parameters in Table 2. The analysis results are shown in Table 7. We see that the parameters μ_H and μ_{VM} assume the greatest importance in system steady state availability of both DCN2 and DCN3, since they present highest absolute values. A major impact upon any change in the value of these parameters bears on the system availability in opposite directions. Sensitivities with respect to these two parameters are negative, since the smaller values the repair times of hosts and VMs get, the higher availability the DCN can achieve. This result reminds the system owner to improve the performance and readiness of the repair and maintenance services in a data center to mitigate recovery time of failed components. Nevertheless, in comparison between the cases of DCN2 and DCN3, the absolute value of the sensitivity with respect to the parameter μ_H is greater in the case of DCN2 than it is in the case of DCN3. However the absolute value of

the sensitivity with respect to the parameter μ_{VM} is higher in the case of DCN3 compared to it in the case of DCN2. The results imply that, in the DCN with more hosts in a $DCell_0$ and more $DCell_0$ units in the network (DCN3 in comparison with DCN2), the recovery of software subsystems (VMs) plays a more important role compared to the recovery of hardware systems (hosts). Thus in the DCell-based DCN with higher number of VMs and $DCell_0$ units, a failure of a host does not cause a significant impact on the operations of a VM compared to the failure of the VM itself, since the VMs have more chances to be migrated onto other hosts in other $DCell_0$s. Therefore, the DCN system designer ought to consider the thorough adoption of software fault tolerance on VMs in a DCN. In Table 7 we also see that the parameter S_{VM} contributes a significant impact on the system availability. The negative values of the sensitivities with respect to the parameter S_{VM} in both cases of DCN2 and DCN3 say that the bigger size of a VM in storage system causes a declining tendency of system availability, since the VM migration processes between hosts within or between $DCell_0$ units last longer to complete. Furthermore, the sensitivity with respect to the parameter ω_m has higher value than it with respect to the parameter ω_{mig}, and both are positive. This is to say that an increase in network speed leads to a corresponding increase of system availability, since the time to migrate VMs could be reduced. Also, the link bandwidth of the pairs of hosts between $DCell_0$s (ω_m) reveals a more important contribution on the system availability than that of the pairs of hosts within a $DCell_0$ (ω_{mig}). The reason is that the cross-links of the hosts between $DCell_0$s are to tolerate the switch failures (which disconnect the communication between system users and VMs in a $DCell_0$) so that a VM is migrated from a $DCell_0$ to others upon any failure of the switch in the $DCell_0$. However, the cross-links could cause the high complexity of network routing and the requirements of high speed links could lead to a huge amount of overall system cost. Thus the system design has to be aware of the trade-offs between system availability and performance and the overall cost of networking.

Figure 7 shows the sensitivity analysis results with respect to the major impacting parameters in both DCN2 and DCN3. The analyses are carried out by altering the value of a parameter of interest as the other parameters remained constant.

Figures 7(a) and 7(b) show the analysis results with respect to MTTFs of host (λ_H), VM (λ_{VM}), and switch (λ_S). There are several similarities of the graphs in which (i) in the early period (0–1000] hours the system availability increases quickly as long as the MTTFs increase and (ii) the system availability slowly increases and approaches a steady value as the MTTFs get greater values in the late period (over 1000 hours). Also, switches in each DCell show its major impact on the system availability. If the MTTF of switches gets a low value in the early period (the switches fails more frequently), the system availability is severely pulled down in comparison with the sensitivity analysis results of system availability with respect to MTTFs of host and VM. The MTTFs of host and VM only contribute a little impact on system availability in the early period (showed by declining vertical graphs with circle and star markers) but mostly do not cause a great impact on system availability in the late period (depicted by approximately horizontal graphs with circle and star markers). This is to say that a DCN is likely prone to switches' failures. Since the switches are the key components to connect a number of physical hosts in $DCell_0$s, a failure of a switch severely causes a failure of the whole $DCell_0$ (unable to connect system user to the $DCell_0$).

Figures 7(c) and 7(d) present the results of availability sensitivity analysis with respect to MTTRs of host (μ_H), VM (μ_{VM}), and switch (μ_S). The figures apparently reflect the significant impact of the MTTR of software system (VMs) onto the overall system availability. In both DCN2 and DCN3, as the MTTR of VMs increases to get higher values, the system availability slides down very quickly depicted by the graph marked with stars. This is because the user's applications run on VMs hence the VMs' up or down states decisively influence the system availability. Moreover, in the DCN2 with less number of hosts, the increase of MTTR of hosts can decrease the system availability as shown by the graph with circle marker in Figure 7(c). But in the DCN3 with more numbers of hosts, the value of MTTR of hosts does not significantly impact the system availability as shown in Figure 7(d). The reason is that if a host fails, the VM running on that host can be migrated to other hosts in the same $DCell_0$. In the DCN2 with less number of hosts, the longer time the repair of hosts spends, the less chance the system can have to be available. In the DCN3 with more numbers of hosts and under the assumption of a high available system where a host can be recovered before the last host's failure, the MTTR of hosts mostly does not impact the system availability. At last, the MTTR of switches does not affect the system availability as depicted by the graphs with triangle markers in Figures 7(c) and 7(d), since, as long as a switch fails, all VMs running on the $DCell_0$ of that switch are migrated to the other $DCell_0$s.

Figures 7(e) and 7(f) depict the availability sensitivity with respect to network bandwidths within a $DCell_0$ (ω_{mig}) and between $DCell_0$s (ω_m). If the network speeds within or between $DCell_0$s surpass a specific value at about 400 Mb/s, the system can achieve high availability. However if the speeds get slower, the system availability is pulled down quickly. The figures also reflect the importance of network bandwidth within a $DCell_0$ compared to that between $DCell_0$s. The low value of the network bandwidth within a $DCell_0$ pulls down the system availability more severely (as depicted by the vertical slope of the star-marked graph) than that between $DCell_0$s does. The reason is that the connection between hosts in a $DCell_0$ is to tolerate hosts' failures which are more frequent to occur but the connection between hosts among different $DCell_0$ is to tolerate switches' failures which happen less frequently.

Figure 7(g) shows the availability sensitivity with respect to VM image sizes (S_{VM}). Under the default values of parameters, the size of VM image files affects the system availability in a negative manner. As the size increases, the system availability slides down quickly. Furthermore, the VM image size has greater influence on the system availability in the DCN with less number of hosts in a $DCell_0$ (DCN2) than that in the DCN with more numbers of hosts (DCN3) does. The bigger size of VM can pull down the system availability

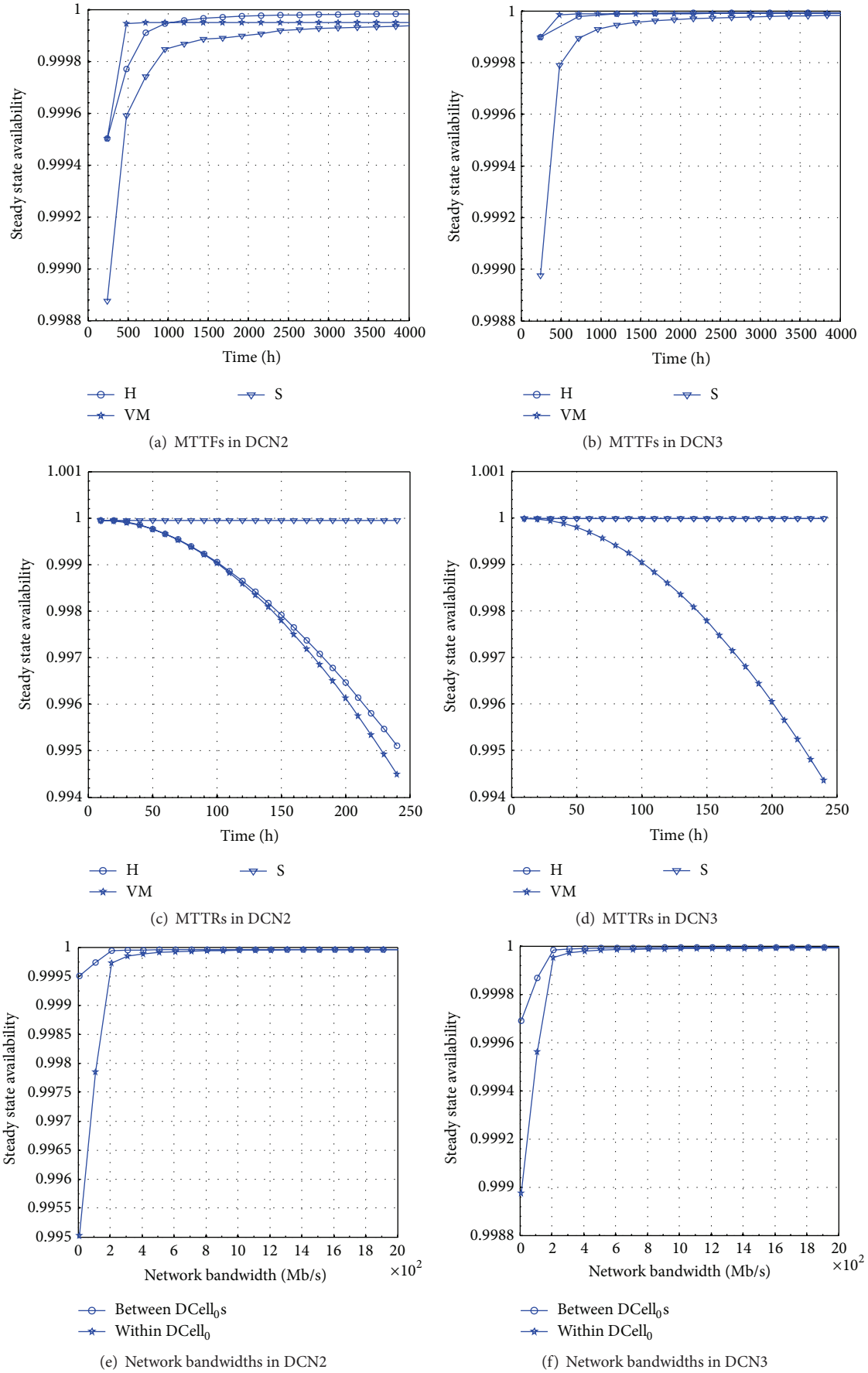
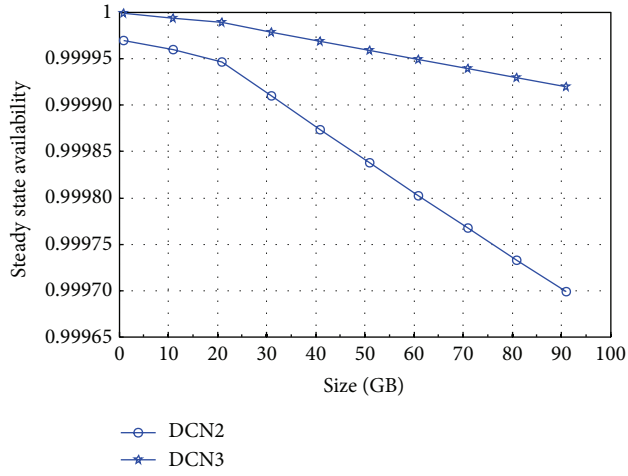


Figure 7: Continued.



(g) VM image sizes

Figure 7: Sensitivity analysis of DCell-based DCNs.

more quickly in DCN2 in comparison with that in DCN3. This is depicted by the smaller slope of the star-marked graph (for DCN3) compared to that of the circle-marked graph (DCN2). This result implies that, in a DCell-based DCN with higher number of physical hosts in a $DCell_0$, the system can have better ability to tolerate hardware and software failures and thus be able to deliver bigger size of VM image files.

5.3. Discussion. A practical DCell-based DCN system comprises tens to hundreds of thousands of hardware components (hosts, switches, links, etc.) and thus hosts even an enormous number of VMs in a very complicated topology of networking and routing as described in [1, 4]. An effort to model and analyze such DCN system is critically important to help provide a guide basis for design and management of both hardware and software subsystems. We find this a fruitful topic for further work on system scalability. Nevertheless, the endeavor to build a complete and monolithic model to capture the whole system behaviors also confronts largeness problem (also known as state-space explosion) in modeling. To deal with this issue, one may adopt different modeling techniques and methodologies such as state truncation [61], state aggregation [62], model decomposition [63, 64], state exploration [65, 66], and model composition [67, 68]. Other different methodologies have been also adopted popularly in literature, which are also appropriate to deal with scalability and largeness problems of modeling a large DCN system such as (i) hierarchical models, which partition a complex model into a hierarchy of submodels [69] or combine combinatorial models and state-space models [70–72], (ii) interactive models [22, 73, 74], which divide a large monolithic model into a number of smaller scale models with comprehensive interactions and dependencies, (iii) fixed-point iterative models [75], and (iv) discrete-event simulation [76]. Thus this is a broad future research avenue to scale up system configuration and to resolve the largeness problem in modeling a DCN system. In this paper, our sharp focus is on system capability of fault tolerance and business continuity through availability modeling and analysis. We have shown that the DCell-based DCN can have higher availability to assure business continuity even in the presence of severe failures of components. Nevertheless, it is necessary to observe and study the system in different perspectives including reliability [77], survivability [78], performability [79] for instance. These topics are still open for future endeavor.

6. Conclusion

This paper has presented a comprehensive availability modeling and sensitivity analysis of a DCell-based DCN. Our work studied two typical DCell configurations of the DCN, respectively, comprising two and three hosts in a $DCell_0$. Our focus is on the fault tolerant capability and business continuity of the DCNs; thus the VM live migration mechanisms are incorporated in detail to tolerate failures of switches and hosts. The modeling captured the distributed fault tolerant routing protocol designed in system architectures. A variety of analyses were carried out thoroughly in consideration of different measures of interest. The steady state availability analyses have shown that the DCell-based DCNs can assure HA and business continuity, tolerate hardware failures of switches and hosts, and enhance vastly the system's overall availability. The increasing number of VMs in a DCN slightly improves the system availability but causes a high complexity and largeness problems in modeling and analysis. The comprehensive sensitivity analyses of system steady state availability were also performed in order to observe the system characteristics and behaviors upon any change of major impacting parameters. The sensitivity analysis results have pointed out that (i) recovery actions of hosts and VMs are significantly important to mitigate system downtime, (ii) recovery actions of software subsystem (VMs) in a DCell-based DCN cause major impacts on system availability in comparison with those of hardware subsystems (hosts and switches), and (iii) network bandwidth of the link between $DCell_0$s is a critical parameter to obtain and maintain high

availability of the system. This study brings about a guide basis to help manage and operate a DCN in data centers in terms of (i) maintenance and repair readiness, (ii) awareness of software fault tolerance in DCNs, and (iii) selection basis of network performance and availability and cost to avoid potential risks as well as tolerate faults.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2015-H8501-15-1011) supervised by the IITP (Institute for Information & Communications Technology Promotion). The authors would like to thank Professor Dr. Kishor Trivedi, Professor of Duke University, United States, for providing SPNP.

References

[1] Y. Liu and J. Muppala, "Fault-tolerance characteristics of data center network topologies using fault regions," in *Proceedings of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '13)*, pp. 1–6, 2013.

[2] D. A. Patterson, "A simple way to estimate the cost of downtime," in *Proceedings of the 16th USENIX Conference on System Administration*, pp. 185–188, 2002.

[3] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63–74, 2008.

[4] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: a scalable and fault-tolerant network structure for data centers," in *Proceedings of the ACM SIGCOMM Conference on Data Communication (SIGCOMM '08)*, pp. 75–86, Seattle, Wash, USA, August 2008.

[5] C. Guo, G. Lu, D. Li et al., "BCube: a high performance, server-centric network architecture for modular data centers," in *Proceedings of the ACM SIGCOMM Conference on Data Communication (SIGCOMM '09)*, pp. 63–74, ACM, Barcelona, Spain, August 2009.

[6] M. Manzano, K. Bilal, E. Calle, and S. U. Khan, "On the connectivity of data center networks," *IEEE Communications Letters*, vol. 17, no. 11, pp. 2172–2175, 2013.

[7] R. Mikkilineni and G. Kankanhalli, "Using virtualization to prepare your data center for 'real-time assurance of business continuity'," in *Proceedings of the 19th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE '10)*, pp. 76–81, June 2010.

[8] K. Bilal, S. U. R. Malik, O. Khalid et al., "A taxonomy and survey on green data center networks," *Future Generation Computer Systems*, vol. 36, pp. 189–208, 2014.

[9] S. Loveland, E. M. Dow, F. LeFevre, D. Beyer, and P. F. Chan, "Leveraging virtualization to optimize high-availability system configurations," *IBM Systems Journal*, vol. 47, no. 4, pp. 591–604, 2008.

[10] T. A. Nguyen, D. S. Kim, and J. S. Park, "A comprehensive availability modeling and analysis of a virtualized servers system using stochastic reward nets," *The Scientific World Journal*, vol. 2014, Article ID 165316, 18 pages, 2014.

[11] M. F. Bari, R. Boutaba, R. Esteves et al., "Data center network virtualization: a survey," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 2, pp. 909–928, 2013.

[12] C. Clark, K. Fraser, S. Hand et al., "Live migration of virtual machines," in *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation (NSDI '05)*, vol. 2, pp. 273–286, USENIX Association, Berkeley, Calif, USA, 2005.

[13] R. W. Ahmad, A. Gani, S. H. A. Hamid, M. Shiraz, A. Yousafzai, and F. Xia, "A survey on virtual machine migration and server consolidation techniques for cloud data centers," *Journal of Network and Computer Applications*, vol. 52, pp. 11–25, 2015.

[14] E. Rodriguez, G. Alkmim, D. M. Batista, and N. L. S. Da Fonseca, "Live migration in green virtualized networks," in *Proceedings of the IEEE International Conference on Communications (ICC '13)*, pp. 2262–2266, June 2013.

[15] Y. Liu, D. Lin, J. Muppala, and M. Hamdi, "A study of fault-tolerance characteristics of data center networks," in *Proceedings of the IEEE/IFIP 42nd International Conference on Dependable Systems and Networks Workshops (DSN-W '12)*, pp. 1–6, June 2012.

[16] K. Bilal, M. Manzano, S. U. Khan, E. Calle, K. Li, and A. Y. Zomaya, "On the characterization of the structural robustness of data center networks," *IEEE Transactions on Cloud Computing*, vol. 1, no. 1, p. 1, 2013.

[17] K. S. Trivedi, D.-S. Kim, and R. Ghosh, "System availability assessment using stochastic models," *Applied Stochastic Models in Business and Industry*, vol. 29, no. 2, pp. 94–109, 2013.

[18] R. D. S. Matos Jr., A. P. Guimarães, K. M. A. Camboim, P. R. M. Maciel, and K. S. Trivedi, "Sensitivity analysis of availability of redundancy in computer networks," in *Proceedings of the 4th International Conference on Communication Theory, Reliability, and Quality of Service (CTRQ '11)*, pp. 115–121, April 2011.

[19] R. Alshahrani and H. Peyravi, "Modeling and simulation of data center networks," in *Proceedings of the 2nd ACM SIGSIM/PADS Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS '14)*, pp. 75–82, May 2014.

[20] ADC Telecommunications, "TIA-942 data center standards overview," White Paper, ADC Telecommunications, Minneapolis, Minn, USA, 2006.

[21] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, p. 68, 2008.

[22] F. Longo, R. Ghosh, V. K. Naik, and K. S. Trivedi, "A scalable availability model for Infrastructure-as-a-Service cloud," in *Proceedings of the IEEE/IFIP 41st International Conference on Dependable Systems and Networks (DSN '11)*, pp. 335–346, June 2011.

[23] R. N. Mysore, A. Pamboris, N. Farrington et al., "PortLand: a scalable fault-tolerant layer 2 data center network fabric," in *Proceedings of the ACM SIGCOMM Conference on Data Communication (SIGCOMM '09)*, pp. 39–50, August 2009.

[24] M. Al-fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: dynamic flow scheduling for data center

networks," in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (NSDI '10)*, p. 19, Boston, Mass, USA, March 2010.

[25] A. Greenberg, J. R. Hamilton, N. Jain et al., "VL2: a scalable and flexible data center network," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, p. 51, 2009.

[26] D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, and S. Lu, "FiConn: using backup port for server interconnection in data centers," in *Proceedings of the 28th Conference on Computer Communications (IEEE INFOCOM '09)*, pp. 2276–2285, April 2009.

[27] D. Lin, Y. Liu, M. Hamdi, and J. Muppala, "Hyper-BCube: a scalable data center network," in *Proceedings of the IEEE International Conference on Communications (ICC '12)*, pp. 2918–2923, June 2012.

[28] T. Lumpp, J. Schneider, J. Holtz et al., "From high availability and disaster recovery to business continuity solutions," *IBM Systems Journal*, vol. 47, no. 4, pp. 605–619, 2008.

[29] P. Patel, A. Ranabahu, and A. Sheth, *Service Level Agreement in Cloud Computing*, Kno.e.sis Publications, 2009.

[30] S. K. Garg, A. N. Toosi, S. K. Gopalaiyengar, and R. Buyya, "SLA-based virtual machine management for heterogeneous workloads in a cloud datacenter," *Journal of Network and Computer Applications*, vol. 45, pp. 108–120, 2014.

[31] T. Thein, S.-D. Chi, and J. S. Park, "Improving fault tolerance by virtualization and software rejuvenation," in *Proceedings of the 2nd Asia International Conference on Modelling & Simulation (AMS '08)*, pp. 855–860, May 2008.

[32] J. Daniels, "Server virtualization architecture and implementation," *Crossroads*, vol. 16, no. 1, pp. 8–12, 2009.

[33] P. Barham, B. Dragovic, K. Fraser et al., "Xen and the art of virtualization," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, vol. 37, p. 164, ACM, Bolton Landing, NY, USA, October 2003.

[34] R. Y. Ameen and A. Y. Hamo, "Survey of server virtualization," *International Journal of Computer Science and Information Security*, vol. 11, no. 3, pp. 65–74, 2013.

[35] M. Melo, P. Maciel, J. Araujo, R. Matos, and C. Araujo, "Availability study on cloud computing environments: Live migration as a rejuvenation mechanism," in *Proceedings of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '13)*, pp. 1–6, June 2013.

[36] P. G. J. Leelipushpam and J. Sharmila, "Live VM migration techniques in cloud environment—a survey," in *Proceedings of the IEEE Conference on Information and Communication Technologies (ICT '13)*, pp. 408–413, April 2013.

[37] J. T. Blake, A. L. Reibman, and K. S. Trivedi, "Sensitivity analysis of reliability and performability measures for multiprocessor systems," in *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '88)*, vol. 16, pp. 177–186, ACM, Santa Fe, NM, USA, May 1988.

[38] R. Matos, J. Araujo, D. Oliveira, P. Maciel, and K. Trivedi, "Sensitivity analysis of a hierarchical model of mobile cloud computing," *Simulation Modelling Practice and Theory*, vol. 50, pp. 151–164, 2015.

[39] M. P. Bendsøe and N. Kikuchi, "Generating optimal topologies in structural design using a homogenization method," *Computer Methods in Applied Mechanics and Engineering*, vol. 71, no. 2, pp. 197–224, 1988.

[40] P. M. Frank and M. Eslami, "Introduction to system sensitivity theory," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 10, no. 6, pp. 337–338, 1980.

[41] R. Y. Al-Jaar, "Book review: the art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling by Raj Jain (John Wiley & Sons 1991)," *ACM SIGMETRICS Performance Evaluation Review*, vol. 19, no. 2, pp. 5–11, 1991.

[42] K. S. Trivedi, D. S. Kim, A. Roy, and D. Medhi, "Dependability and security models," in *Proceedings of the 7th International Workshop on the Design of Reliable Communication Networks*, pp. 11–20, October 2009.

[43] T. Wang, Z. Su, Y. Xia, Y. Liu, J. Muppala, and M. Hamdi, "SprintNet: a high performance server-centric network architecture for data centers," in *Proceedings of the 1st IEEE International Conference on Communications (ICC '14)*, pp. 4005–4010, June 2014.

[44] X. Wu, D. Turner, G. Chen et al., "NetPilot: automating datacenter network failure mitigation," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 443–454, 2012.

[45] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," in *Proceedings of the ACM SIGCOMM Conference*, pp. 350–361, ACM, Ontario, Canada, August 2011.

[46] K. V. Vishwanath and N. Nagappan, "Characterizing cloud computing hardware reliability," in *Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC '10)*, pp. 193–204, June 2010.

[47] M. G. Rabbani, M. F. Zhani, and R. Boutaba, "On achieving high survivability in virtualized data centers," *IEICE Transactions on Communications*, vol. E97-B, no. 1, pp. 10–18, 2014.

[48] T. Adeshiyan, C. R. Attanasio, E. M. Farr et al., "Using virtualization for high availability and disaster recovery," *IBM Journal of Research and Development*, vol. 53, no. 4, pp. 8:1–8:11, 2009.

[49] E. Bin, O. Biran, O. Boni et al., "Guaranteeing high availability goals for virtual machine placement," in *Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS '11)*, pp. 700–709, July 2011.

[50] D. S. Kim, F. Machida, and K. S. Trivedi, "Availability modeling and analysis of a virtualized system," in *Proceedings of the 15th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC '09)*, pp. 365–371, November 2009.

[51] W. E. Smith, K. S. Trivedi, L. A. Tomek, and J. Ackaret, "Availability analysis of blade server systems," *IBM Systems Journal*, vol. 47, no. 4, pp. 621–640, 2008.

[52] J. K. Muppala and C. Lin, "Dependability analysis of large-scale distributed systems using stochastic Petri nets," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 4, pp. 3033–3038, IEEE, Beijing, China, October 1996.

[53] F. Machida, D. S. Kim, and K. S. Trivedi, "Modeling and analysis of software rejuvenation in a server virtualized system with live VM migration," *Performance Evaluation*, vol. 70, no. 3, pp. 212–230, 2013.

[54] Y. Cao, H. Sun, K. S. Trivedi, and J. J. Han, "System availability with non-exponentially distributed outages," *IEEE Transactions on Reliability*, vol. 51, no. 2, pp. 193–198, 2002.

[55] G. Ciardo, J. K. Muppala, K. S. Trivedi, J. K. Muppala, G. Ciardo, and K. S. Trivedi, "Stochastic reward nets for reliability prediction," *Journal of Parallel and Distributed Computing*, vol. 15, no. 3, pp. 255–269, 1992.

[56] C. Constazltinescu and K. Trivedi, "A stochastic reward net model for dependability analysis of real-time computing systems," in *Proceedings of the 2nd IEEE Workshop on Real-Time Applications*, pp. 142–146, Washington, DC, USA, 1994.

[57] G. Ciardo, J. Muppala, and K. S. Trivedi, "SPNP: stochastic Petri net package," in *Proceedings of the 3rd International Workshop on Petri Nets and Performance Models (PNPM '89)*, pp. 142–151, IEEE, Kyoto, Japan, December 1989.

[58] B. Silva, P. Maciel, E. Tavares, and A. Zimmermann, "Dependability models for designing disaster tolerant cloud computing systems," in *Proceedings of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '13)*, pp. 1–6, June 2013.

[59] F. Machida, D. S. Kim, and K. S. Trivedi, "Modeling and analysis of software rejuvenation in a server virtualized system," in *Proceedings of the IEEE 2nd International Workshop on Software Aging and Rejuvenation*, pp. 1–6, IEEE, San Jose, Calif, USA, November 2010.

[60] M. Stansberry, *2013 Data Center Industry Survey*, Uptime Institute LLC, New York, NY, USA, 2013.

[61] K. Goševa-Popstojanova and K. S. Trivedi, "Stochastic modeling formalisms for dependability, performance and performability," in *Performance Evaluation: Origins and Directions*, vol. 1769 of *Lecture Notes in Computer Science*, pp. 403–422, Springer, Berlin, Germany, 2000.

[62] P. Buchholz, "An adaptive aggregation/disaggregation algorithm for hierarchical Markovian models," *European Journal of Operational Research*, vol. 116, no. 3, pp. 545–564, 1999.

[63] S. M. Koriem, "Accurate approximate analysis of cell-based switch architectures," *Journal of Systems and Software*, vol. 45, no. 2, pp. 155–171, 1999.

[64] M. A. Marsan, R. Gaeta, and M. Meo, "Accurate approximate analysis of cell-based switch architectures," *Performance Evaluation*, vol. 45, no. 1, pp. 33–56, 2001.

[65] I. Davies, W. J. Knottenbelt, and P. S. Kritzinger, "Symbolic methods for the state space exploration of GSPN models," in *Proceedings of the 12th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation (TOOLS '02)*, pp. 188–199, 2002.

[66] A. S. Miner, "Efficient state space generation of GSPNs using decision diagrams," in *Proceedings of the International Conference on Dependable Systems and Networks (DNS '02)*, pp. 637–646, June 2002.

[67] P. Ballarini, S. Donatelli, and G. Franceschinis, "Parametric stochastic well-formed nets and compositional modelling," in *Application and Theory of Petri Nets 2000: 21st International Conference, ICATPN 2000 Aarhus, Denmark, June 26–30, 2000 Proceedings*, vol. 1825 of *Lecture Notes in Computer Science*, pp. 43–62, Springer, Berlin, Germany, 2000.

[68] W. J. Knottenbelt, P. G. Harrison, M. A. Mestern, and P. S. Kritzinger, "A probabilistic dynamic technique for the distributed generation of very large state spaces," *Performance Evaluation*, vol. 39, no. 1–4, pp. 127–148, 2000.

[69] M. Lanus, L. Yin, and K. S. Trivedi, "Hierarchical composition and aggregation of state-based availability and performability models," *IEEE Transactions on Reliability*, vol. 52, no. 1, pp. 44–52, 2003.

[70] O. C. Ibe, R. C. Howe, and K. S. Trivedi, "Approximate availability analysis of VAXcluster systems," *IEEE Transactions on Reliability*, vol. 38, no. 1, pp. 146–152, 1989.

[71] K. S. Trivedi, R. Vasireddy, D. Trindade, S. Nathan, and R. Castro, "Modeling high availability," in *Proceedings of the 12th Pacific Rim International Symposium on Dependable Computing (PRDC '06)*, pp. 154–164, December 2006.

[72] J. T. Blake and K. S. Trivedi, "Reliability analysis of interconnection networks using hierarchical composition," *IEEE Transactions on Reliability*, vol. 38, no. 1, pp. 111–120, 1989.

[73] R. Ghosh, F. Longo, V. K. Naik, and K. S. Trivedi, "Modeling and performance analysis of large scale IaaS clouds," *Future Generation Computer Systems*, vol. 29, no. 5, pp. 1216–1234, 2013.

[74] R. Ghosh, K. S. Trivedi, V. K. Naik, and D. S. Kim, "End-to-end performability analysis for infrastructure-as-a-service cloud: an interacting stochastic models approach," in *Proceedings of the 16th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC '10)*, pp. 125–132, December 2010.

[75] V. Mainkar and K. S. Trivedi, "Fixed point iteration using stochastic reward nets," in *Proceedings of the 6th International Workshop on Petri Nets and Performance Models*, pp. 21–30, October 1995.

[76] T. Angskun, G. Bosilca, G. Fagg, J. Pješivac-Grbović, and J. J. Dongarra, "Reliability analysis of self-healing network using discrete-event simulation," in *Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid '07)*, pp. 437–444, May 2007.

[77] E. Topuz, "Reliability and availability basics," *IEEE Antennas and Propagation Magazine*, vol. 51, no. 5, pp. 231–236, 2009.

[78] J. F. Castet and J. Saleh, "Survivability and resiliency of spacecraft and space-based networks: a framework for characterization and analysis," in *Proceedings of the AIAA SPACE Conference & Exposition*, San Diego, Calif, USA, September 2008.

[79] R. Entezari-Maleki, K. S. Trivedi, and A. Movaghar, "Performability evaluation of grid environments using stochastic reward nets," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 2, pp. 204–216, 2015.