This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access

Digital Object Identifier

Reliability and Availability Evaluation for Cloud Data Center Networks using Hierarchical Models

TUAN ANH NGUYEN[1,2], DUGKI MIN[2], EUNMI CHOI[3], and TRAN DUC THANG[4]
[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
[4]Institute of Information Technology, Vietnam Academy of Science and Technology, Hanoi, Vietnam
Email: {anhnt2407, dkmin}@konkuk.ac.kr, emchoi@kookmin.ac.kr, thang@ioit.ac.vn

Corresponding author: Tuan Anh Nguyen (e-mail: anhnt2407@konkuk.ac.kr)
Principal corresponding author: Dugki Min (e-mail: dkmin@konkuk.ac.kr)

ABSTRACT Modeling a cloud computing center is crucial to evaluate and predict its inner connectivity reliability and availability. Many of previous studies on system availability/reliability assessment of virtualized systems consisting of singular servers in cloud data centers have been reported. In this paper, we propose a hierarchical modeling framework for reliability and availability evaluation of tree-based data center networks. The hierarchical model consists of three layers, including (i) reliability graphs in the top layer to model the system network topology, (ii) a fault-tree to model the architecture of the subsystems, and (iii) stochastic reward nets to capture the behaviors and dependency of the components in the subsystems in detail. Two representative data center networks based on three-tier and fat-tree topologies are modeled and analyzed in a comprehensive manner. We specifically consider a number of case-studies to investigate the impact of networking and management on cloud computing centers. Furthermore, we perform various detailed analyses with regard to reliability and availability measures for the system models. The analysis results show that appropriate networking to optimize the distribution of nodes within the data center networks can enhance the reliability/availability. The conclusion of this study can be used toward the practical management and construction of cloud computing centers.

INDEX TERMS Data Center Network (DCN), Reliability, Availability, Hierarchical Modeling, Reliability Graph (RG), Fault Tree (FT), Stochastic Reward Net (SRN)

I. INTRODUCTION

In modern ICT ecosystems, data center (DC)s play the role of a centric core. The huge network system of physical servers in DCs (also known as the data center network (DCN) [1]) facilitates the continuous operation of online businesses and information services from distant parts of the world. Under strict requirements to mitigate any catastrophic failures and system outages, DC systems are in the progress of rapid expansion and redesign for high reliability and availability [2]. The reliability/availability of a certain server system in DCs is commonly supposed to be dependent on the reliability/availability of its own physical subsystems as well as the number of subsystems involved in the system architecture. However, because every compute node in a DCN communicates with other nodes via a network topology, it is a matter of curiosity that different manipulations of a certain system with similar components can gain different measures of interest. Thus, even though the number of components remains unchanged, their appropriate allocation and networking can significantly improve the reliability/availability of the system. Few studies on the extent to which the allocation and interconnection of subsystems can affect the reliability/availability of the overall system in DCNs have been published.

An appropriate architecture to interconnect the physical servers in a DCN is important for the agility and reconfigurability of DCs. The DCNs are required to respond to heterogeneous application demands and service requirements with high reliability/availability as well as high performance and throughput. Contemporary DCs employ top of rack (ToR) switches interconnected through end of rack (EoR) switches, which are, in turn, connected to core switches.

Nevertheless, recent studies proposed a variety of network topology designs in which each approach features its unique network architecture, fault avoidance and recovery, and routing algorithms. We adopt the architecture classification of DCN presented in [3] to categorize DCNs into three main classes: *(i) switch-centric architectures*, for instance, Three-tier [4], Fat-Tree [5], PortLand[6], and F^2Tree[7]; *(ii) server-centric architectures* (also known as recursive topologies [8]) e.g, DCell[9], Ficonn[10], MCube[11], and *(iii) hybrid/enhanced architectures*, e.g., Helios[12]. In practice, four main network topologies are widely used to construct server networks in DCs including two switch-centric topologies (three-tier and fat-tree), and two server-centric topologies (BCube, DCell). Among these topologies, fat-tree (and its variants) is a potential candidate of DCN topologies for mass-built DCs of giant online-business enterprises such as Google [13] and Facebook [14]. The use of a large number of small, commodity and identical switches help reduce the construction budget for a new DC significantly while balancing other measures and characteristics of a DCN [5]. The small and identical switches differ only in their configuration and placement in the network, but they deliver low power bandwidth operational expenditure (OPEX) and capital expenditure (CAPEX). Furthermore, the deployment of pods in fat-tree topology can be incremental without any downtime or rewiring when the size of DC is requested to scale/built out. Also, network softwares are not required to be written to be network aware when considering a good performance, which is the biggest advantage of fat-tree topology [15]. Cabling complexity is, however the daunting disadvantage of the fat-tree topology in practical deployment. In comparison to other relevant DCN topologies, fat-tree outperforms in various measures. For instance, fat-tree is better than DCell and BCube in terms of some performance-related metrics such as throughput and latency [13]. In comparison with three-tier topology, fat-tree DCNs do not require the use of high-end switches and high-speed links, thus can drop the total deployment cost rapidly [5]. In general, the common metrics to assess a DCN in practice are scalability, path diversity, throughput and latency, power consumption, and cost [16]. More recently, to maintain long-running online services, the ability of DCNs to tolerate multiple failures (of links, switches and compute nodes) is an essential characteristic requiring urgent consideration for DCNs [8]. Thus, appropriate modeling and evaluation of the fault-tolerance characteristics using stochastic models are necessary to enhance the reliability/availability for DCNs. In this paper, we focus on exploring fault-tolerant indicators of connectivity in a DCN including reliability/availability for the simplest non-trivial instance of fat-tree topology (as a widely-used candidate in industry) in comparison with three-tier topology (contemporarily used in many giant DCs) using stochastic models.

A failure of network elements in DCNs is inevitable. Therefore, the network requires automatic reconfiguration mechanisms and restoration of network services at the moment of failure until a complete repair of the faults of nodes/links becomes possible. Service outages due to any type of failures in a DC significantly incur huge costs on both providers and customers. A study carried out by Ponemon Institute [17] among 63 DCs shows that, the average cost since 2010 due to downtime of each DC has increased 48% from 500,000USD to 740,357USD. In addition, according to a report [18] on failure rates within the Google clusters of 1,800 physical servers (used as building blocks in the IT infrastructure of Google Data Centers), there are roughly 1,000 individual machine failures and thousands of hard drive failures in each cluster during the first year of operations, also the cost to repair each failure reaches almost 300USD, not considering the losses caused directly by the failure in terms of operational business revenues. Thus, reliability/availability evaluation of a cloud-based DC requires a comprehensive model in which different types of failures and factors causing the failures are necessarily taken into account. The detailed analysis of such models could also help technicians to choose appropriate routing policies in the deployment of IT infrastructure.

In this paper, we consider an important physical infrastructure in DCs for securing the continuous operations of data processing in a cloud computing system, which is a network of servers (namely, DCN). To secure the operational continuity in a DC, it demands to prolong at the highest level of reliability/availability of network connectivity and physical subsystems. As discussed in [19], reliability/availability are essential metrics in business-related assessment processes of a computing system for high availability (HA) and business continuity. In cloud DCs, data-intensive processing tasks and constantly online business services often require highly reliable and available connectivity between compute nodes. Therefore, the DCs for business continuity of cloud computing services demand a comprehensive assessment in a complete manner at all levels of the infrastructure.

The use of the term cloud DC is to emphasize on high availabiliy and business continuity factors of the physical infrastructure for constantly online services and data-intensive processing tasks. The infrastructure could be at all sizes from schools to enterprises. We focus on the physical infrastructure in a DC that mainly provide continuous cloud services rather than other infrastructures that are to operate the whole DC as those were discussed in [20]. For the sake of business continuity of a cloud infrastructure, reliability and availability are apparently significant indicators in the evaluation process of system design to assure that the designed infrastructure in a cloud DC would provide the highest level of quality services in accordance with service level agreement (SLA) between the system's owner and cloud end-users. As discussed in [21], availability of an infrastructure in DCs is the probability that the system functions properly at a specific instant or over a predefined period of time. And reliability is the probability that the system functions properly throughout a specific interval of time. Therefore, when we consider reliability, we often take into account failure modes, whereas when

we consider availability, both failure modes and recovery operations are taken into account. These concepts are also applicable for the system in this study in the way that reliability/availability of DCNs represent the connectivity and continuity of the network and services running on top of the infrastructure.

To evaluate the dependability (reliability, availability, performability, etc.) of a certain system, the use of mathematical models, which normally includes state-space models and hierarchical models, is usually an appropriate approach. State-space models (e.g., continuous time Markov chain (CTMC), stochastic petri net (SPN), and stochastic reward net (SRN)) are often used to model the systems that run throughout various operational states with sophisticated dependences between system components. Therefore, a state-space modeling approach can capture the complication of different operational states and processes in a specific system. This is the reason for usually using a state-space modeling approach to model every operational detail of a system. Nevertheless, state-space models are apparently and adversely affected by the state-space explosion problem in most cases, in which the state-space of the constructed model becomes excessively complicated or large to compute and analyze by normal computational solutions. Because of this problem, the drawback of the state-space modeling approach is that state-space-based modeling of the overall system architecture is troublesome and the system model is usually intractable for further analyses. One of the solutions to avoid the state-space explosion problem [22] is to split a large and monolithic state-space based model into different independent sub-models. Each of the individual models is solved and analyzed in separate manner. The analysis outputs of the sub-models are then transferred up to the overall system model. Thus, this approach reduce the sophistication as well as the largeness of the solution for the complete system model, therefore reduce the total computation time. This is the approach of hierarchical modeling.

A number of papers on the presentation and description of DCN topologies have been published [5, 9]. Some other work concerned on different aspects of DCN including fault tolerance characteristics [8], structural robustness of DCN topologies [23] or connectivity of DCNs [24]. Another paper [25] evaluated the reliability and survivability of different DCN topologies based on failures without repairs using the graph manipulation tool NetworkX [26]. Nevertheless, none of these papers presented a quantitative assessment of system behaviors using stochastic models [27]. To the best of our knowledge, only a single recent paper [28] presented thorough performance modeling and analysis of a fat-tree-based DCN using queuing theory. Thus, we found that modeling and analyzing a virtualized DCN with the use of stochastic models with regard to various failure modes and recovery strategies in a complete manner remains a preliminary endeavor. This motivated us to model and analyze tree-based DCNs (three-tier and fat-tree topologies) using a hierarchical modeling framework.

The main contributions of this paper are summarized as follows:

- We proposed a three-layer hierarchical modeling framework specifically for the availability/reliability evaluation of DCNs. The framework is composed of (i) a reliability graph (RG) in the top layer for the modeling of the whole system architecture, (ii) fault tree (FT) in the middle layer to capture reliability/availability characteristics of the subsystems and (iii) SRN models in the lower layer to capture operational behaviors of components;
- We proposed to construct hierarchical and heterogeneous models to not only capture the overall network topologies (which are featured for DCNs), but also to consider the detailed configurations of their subsystems as well as to comprehensively incorporate different operational states (failure modes and recovery behavior) of the lowest component units of two representative DCNs based on three-tier and fat-tree topologies;
- We performed comprehensive analyses and evaluation of different metrics of interest including reliability and availability for typical DCNs based on three-tier and fat-tree network topology.

The modeling and analysis enabled us to draw some conclusions for three-tier and fat-tree DCNs:

- The dispersion of compute nodes in processing tasks in the network improves the availability/reliability of the DCN connectivity within the network, thus secure continuity and HA of data transactions and processes in DCs.
- In a comparison, three-tier routing topology bring about more connectivity opportunities for highly reliable and available data transactions than the fat-tree routing topology does in specific cases.
- Physical servers with their typical properties of failure and recovery have a sensitive impact on the reliability/availability of the whole system.
- In addition, the links between the system components need appropriate consideration to maintain high availability for the system.
- The DCN connectivity considered in this paper exhibits high reliability with more than four nines of availability [29].

To the best of our knowledge, this paper presents a comprehensive and complete modeling and evaluation of the reliability/availability of a computer network system from the network topology down to the system components using a hierarchical modeling manner at a very early stage of current research on such systems.

The structure of this paper is organized in sections as follows. Section I introduces the necessity and novelty of this work. Section II presents comprehensive discussions on the reliability/availability assessment in most related works. Section III describes the modeling framework of this study with a modeling example for easy understanding. Section IV

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access

details the system and scenarios under study. In Section V, hierarchical models are described in detail. Section VI shows numerical results. Lastly, the conclusion and discussion are presented in Section VII.

II. RELATED WORK

Reliability and availability quantification:

Reliability and availability quantification is an essential phase in a system development in order to assess those prominent dependability indicators of a physical cloud infrastructure representing the high quality of service which a cloud provider delivers to cloud users [21]. A certain cloud provider often offers high quality services conforming with a prescribed SLA which specifies quality level indices of a physical infrastructure in a cloud DC [30]. Over the last few years, many efforts have been devoted to quantify reliability and availability indices of physical systems in a cloud DC. In [31], Smith *et al.* presented a comprehensive availability evaluation for a commercial and high-availability server system with multiple physical components namely, IBM BladeCenter®, consisting of 14 seperate blade servers along with necessary supporting subsystems such as shared power supply and cooling subsystems.The study identified availability bottlenecks, evaluated different configurations, thus compared different designs, and demonstrated that the modular blade system designs can deliver nearly five-9s hardware availability to meet customer requirements. The quantitative evaluation of datacenter infrastructure availability was extensively performed in [32] in which a non-exponential failure time distribution was taken into account based on the stochastic characterization of midplane reliability through statistic measurements. Some other works considered a cloud DC as a whole consisting of three main infrastructures including IT, cooling and power to assess the reliability and availability along with other related indices such as sustainability and operational cost of a whole DC [33]. Beside the above reliability and availability quantification for physical systems in a cloud DC, there are also a number of works on the reliability and/or availability quantification for software systems integrated on hardware systems in a cloud DC. Kim *et al.* in [34] presented a detailed quatification of availability index for a virtualized system of two physical servers. The study took into consideration both physical hardware and software subsystems of a server (e.g., OS, CPU, RAM, etc.) associated with detailed representation of their operational states. The availability quantification of a virtualized server system was extended in [35] by considering more sophisticated failure and recovery behaviors of the virtualized software subsystem running on a physical twin servers system. Some works investigated availability of specific small-sized system architectures for a certain funtionality in a cloud DC. Melo *et al.* in [36] quantified availability for a data synchronization server infrastructure performing data synchronization activities between a small-sized server system with other terminals. In [37, 38], the availability characteristics of different private cloud infrastructures with a certain number of clusters based on Eucalyptus platform were explored under a variety of fault tolerance strategies such as standby replication mechanisms or software aging avoiding techniques. Costa *et al.* in [39] quantified availability for a mobile backend as a service platform (namely, MBaaS OpenMobster platform) linking a data storage system in a cloud DC to real-world mobile devices, which is in order to indentify the critical service component in the overall architecture design. Some other recent works explored reliability/availability related issues of a cloud DC system featured with a network inter-connection among distributed physical DC. Hou *et al.* in [40] proposed a service degradability framework for a typical configuration of optical fiber network interconnecting two geographically distributed DC to enhance performance on maximizing the network's service availability. Yao *et al.* [41] explored novel algorithms to optimize backup services associated with an inter-network of DCs by finding the optimal arrangement of backup pairs and data-transfer paths under a certain configuration of the inter-network of DCs, which is to imply that the manipulations of network configuration to optimize a certain service delivery can actually obtains higher indices representing quality of services of a network. Many other works investigated DCNs in different perspectives such as cost-effective and low-latency architecture [42], energy-aware issues [43] or structural robustness [23]. But, very few works charaterized operational failure and recovery behaviors in a detailed manner, thus quantified reliability/availability of server networks in cloud DCs. Liu *et al.* in [42] explored the effects of correlated failure behaviors in DCNs captured through the use of fault regions, which is the case of a set of connected components failing together. The study considered different metrics of interest including bottleneck throughput, average path length and routing failure rate. However, reliability/availability were not considered and quantified in an adequate manner. Alshahrani *et al.* in [28] presented a detailed analytical modeling methodology based on queueing theory, however, to evaluate performance indices (throughput and delay) of a typical fat-tree based DCN. Couto *et al.* in [44] presented a preliminary study on only reliability of network topologies in cloud DCs. Nevertheless, the study only took into account the failures of the main network elements (servers, switches, and links) as failure nodes in an undirected graph without paying a proper consideration on repair behaviors and other related failure causes and operational states of the underlying subsystems for a comprehensive quantification of reliability and availability. Our previous works presented preliminary studies on availability quantification for different types of network in a cloud DC. In the work [45], we presented a comprehensive availability quantification of a DCell-based DCN taken into account virtual machine (VM) migration techniques as the main fault-tolerant methodology to enhance the overall system availability. This study considered only two-states representation for physical entities including servers and switches, and focused on capturing detailed

operational behaviors and interaction in the virtualized layer of VMs. In the work [46], we quantified availability of a physical software defined network (SDN) infrastructure complying with a simple network topology. We elaborated more detailed operational failure and recovery behaviors of the physical entities in the network, such as switches and storages. Due to flexibility requirements in forming network topologies in a SDN, we demonstrated that the variation of connection manipulations in the SDN obtain different availabilities of network connectivity and overall system operation, thus, suggested a proper SDN network management to gain optimal metrics of interest. In this paper, we present an extensive study to our previous works by proposing a comprehensive modeling and evaluation framework for a certain network of physical servers in a cloud DC. To the best of our knowledge, the previous work rarely quantified reliability/availability of a network of servers in a cloud DC in a complete manner taking into account either detailed operational state transitions of lowest-level components or manipulations of highest-level network topologies. The requirements of delivering HA services in cloud DCs demand a comprehensive investigation on the impacts of network topology manipulation at the top level of the network as well as the dynamically operational transitions of the involved components at the bottom level of the network, because any change in operations of a single element in the network may cause a huge variation of the sensitive metrics of interest (reliability/availability) in a HA cloud DC. We see this a leading motivation to conduct the study of reliability/availability quantification framework for a DCN in cloud DCs in this paper.

Reliability and availability modeling techniques in practice:

Assuring a high level of business continuity in a cloud DC demands the capability in a system design phase to build-in HA techniques and to differentiate availabilities in the sixth decimal place [19, 47]. Thus, the evaluation of system designs to perform design trade-offs requires a fairly detailed development of stochastic models. Many recent studies have shown different methodologies to develop analytical stochastic models for reliability and availability quantification of different systems. The classification of analytical stochastic models for the quantification of dependability metrics of interest in the previous work goes into the main categories which are usually used in practice as described in the following.

- *Non-state space models* (also known as combinatorial models) such as FT, RG, and reliability block diagram (RBD) allows relatively quick quantification of system reliability and availability with the assumption of statistical independence [48]. FT provides a structured approach using a graphical tree representation of basic events causing a system failure. But, FT only captures a single event of system failure as a top event, therefore if a different type of system in a network, or more generally a system of systems, involves in the modeling, additional FTs must be constructed. FT, therefore is usually helpful in modeling an individual system to intuitively represent the system failure causes in accordance with the system architecture. RG has been extensively used for network reliability quantification [49], which is an acyclic graph with two special nodes labeled, source node (S) which has no coming edges, and sink (D) which has no outgoing edges. The edges in a RG represent the elements of the network to be modeled. The system which is modeled by a RG is considered reliable (operational) if at least one path with no failed edge from source (S) to sink (D) is found. With an intuitive graphical representation, RG is useful in quantifying reliability/availability of networks. Because RGs can be neasted if other models are integrated in the edges, the modeling via RG can capture scalability from simple to highly complex systems. RBDs provide an alternative graphical modeling approach when availability-related system dependencies are taken into consideration. In comparison to other combinatorial models, RG has a good modeling power for complex networks in practice over the other modeling tools. RG models were used to evaluate reliability of various network such as dodecahedron network and 1973 Arpanet networks [50]. In the patent [51], Ramesh *et al.* presented novel reliability estimation methods using RG and applied for the reliability assessment of a current return network in Boeing airplanes, which is a large networked system. Accordingly, techniques to compute reliability/availability upper and lower bounds can help solve extremely large-scale non-state-space models. Our focus in this study of reliability/availability quantification for DCNs is on proposing a framework with the use of nested FTs in an overall RG representing a network of heterogeneous systems.
- *State-space models* are usually used to model complex interactions and behaviors within a system. Under appropriate assumptions, state-space models are also useful in modeling a large-scale system when considering specific behaviors with repetition throughout the large system. A variety of state-space modeling techniques were used in different cases to model various systems in previous work. Markov chain models consisting of *state(s)* and *state transition(s)* are often used to quantify different dependability metrics of interest such as availability along with performance. A Markov chain consisting all transitions labeled with rates is called CTMC. If its transition is labeled with a probability, the model is discrete time markov chain (DTMC). In the cases that a distribution function is used for transition labels, the model is a semi-Markov process (SMP) or markov regenerative stochastic process (MRGP). Thein *et al.* in [52] used CTMC to quantify availability for a dual physical machine system with virtualization. In [53], the study was extended with the incoporation of software rejuvenation technique along with software

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access

virtualization, which were all modeled using CTMC. Matos *et al.* in [54] used CTMC to model VM subsystem and performed detailed sensitivity analyses of availability and operational cost for a dual virtualized servers system (VSS). A SMP were used to quantify availability of a cluster system with disaster recovery in [55]. When a reward is associated with a Markov chain model for the sake of computing a certain metric of interest, the model is also known as markov reward model (MRM). In [56], Trivedi *et al.* used a SMP with general failure and repair distributions to analyze the behaviors of periodic preventive maintenance for the system availability improvement. Other representatives of state-space models that can help ease the modeling of a complex system are Petri net (PN)-based models such as SPN, SRN and fluid stochastic Petri net (FSPN). A PN is a directed bipartite graph with two main elements including *places* and *transitions*. If *tokens* are associated with places, the PN is marked. The transition and allocation of tokens across the PN capture dynamic behaviors of the system to be modeled [57]. If all transitions are associated with exponential distribution of firing time, the PN is SPN. generalized stochastic Petri net (GSPN) allows to have immediate transitions (with zero firing times) and timed transitions (with exponentially distributed firing times). Some other extensions were introduced by Ciardo *et al.* in [58]. In the case when a reward rate is associated with each marking of the net thus, many aspects of a system could be expressed by Boolean expressions and arithmetic involving reward rates, the model is SRN, which was introduced in [59]. Thanks to many extensions in comparison to its predecessors, SRN has a modeling power to capture complex behaviors of a system in a comprehensive manner while the model's size and complexity are reduced significantly. Han *et al.* in [60] developed a SRN model to explore dynamic behaviors of VM migration techniques used in a SDN infrastructure with limited number of physical servers and network devices. The metrics of interest are the availability of VM subsystem that the SDN can deliver and availability-aware power consumption of the whole system. Machida *et al.* in [61] proposed comprehensive SRN models for various time-based virtual machine monitor (VMM) rejuvenation techniques to quantify availability and transactions lost in a year of a VSS. The models take into account a variety of sophisticated system behaviors when a certain software rejuvenation (SRej) technique is incorporated in a compact model thanks to the modeling power of SRN. Yin *et al.* in [62] developed detailed SRN models translated from pre-designed SysML activity diagrams of data backup and restore operation to investigate storage availability, system availability and user-perceived availability in an IT infrastructure. Torquato *et al.* in [63] presented various SRN models to investigate the improvement on availability and power consumption when applying VMM rejuvenation techniques enabled by VM live-migration in a private cloud. In our previous works [21, 35, 64, 65], we developed a number of comprehensive SRN models to capture complex operations in different systems in a cloud DC. Since SRN helps comprehend sophistication of system operations and ease the capturing of the system behaviors expected to investigate [66], we develop SRN models for every individual components in each physical system of specific DCNs in this study.

- *Hierarchical models* (also known as multi-level models) are to avoid largeness problems (also known as state-space explosion problems) which are inherent in modeling large-sized and complex systems and/or multi-level system of systems [67]. The upper levels of an analytical hierarchical model are typically non-state-space model types (for instance, RG for network modeling, FT or RBD for structured modeling of individual systems, as presented in this study), whereas in the lower level, state-space models such as CTMC or SRN are used to capture complex operational behaviors of subsystems or individual components. There are a number of works on the reliability/availability quantification of such sophisticated systems in DCs using multi-levels hierarchical models. [34] was one of the first studies on availability quantification for a VSS in which a hierarchical model was developed consisting of a system FT at the upper level and at the lower level, a number of CTMC models were developed in accordance with different subsystems (including physical hardware and software subsystems) in a server. In [31], a two-level hierarchical model of FT and CTMC were also developed in the same manner to quantify a system of blade servers, in which the FT model corresponds to the overall system of multiple blade servers (without considering a network), whereas CTMC models correspond to physical subsystems (such as server, cooling device, chassis, etc.). In a recent work [68], Lira *et al.* presented an automated strategy using RBD at the upper level and SRN at the lower level in a hierarchical representation for reliability/availability assessment of virtual networks. The approach involves nodes and links along with fault-tolerant techniques among the nodes in the hierarchical modeling. In the work [69], Silva *et al.* proposed an integrated modeling environment contemplating RBD and both SRN and CTMC in a hierarchical manner. A few number of previous works considered more higher hierarchical models for systems with higher level of complexity. Trivedi *et al.* in [70] was one of the first papers that presented a three-level hierarchical modeling approach mixing RBD and Markov chains to develop an availability model for a HA platform in telecommunications industry. The study suggests a specific methodology to develop three-level hierarchical stochastic models for highly available and complex systems. In the work [71], modeling power in hierarchy of both combinatorial

and Markov model types was elaborated in order to recommend appropriate models when assessing a certain system. In particular, even though the non-state-space models are possibly interchangeable at the top level, RG is more powerful than RBD and FT in capturing network topologies. Whereas, the state-space models are also interchangeable at lower levels when involving more sophisticated operations of subsystems and components in a certain system. In general, due to largeness problem, a complex system of multi-levels requires an appropriate modeling methodology of multi-levels in a hierarchical manner in order to cover the overall system architecture at the top level and capture comprehensively system operations at the lowest level. The combination of non-state-space models (at upper levels) and state-space models (at lower levels) in a hierarchy is a proper solution for the above demand. Though the models in the same type are likely inter-changeable [71], we may use specific types of model for certain requirements in modeling, for instance, RG has a proper modeling power and an intuitive representation as we consider a network to be modeled as shown in [51]; FT is an appropriate model type when considering various failure modes at different system levels as shown in [31, 34]; and SRN can capture operational state transitions at the lowest level in a very detailed manner but still maintain the tractability and reduced-size of the model. Therefore, we find that the combination of those models in a hierarhical manner could be a proper solution for the sake of modeling complex systems in a DC. Thanks to the capability to model different system levels, the multi-level hierarchical modeling methodology is practically useful in reliability/availability assessment of a system of systems and/or a network of systems. This paper advocates a three-level hierarchical modeling framework for a network of systems, specifically a network of servers in DCs. We attemps to apply the assessment framework for a typical DCN at the greatest level of detail while balancing the models' complexity.

III. A HIERARCHICAL MODELING FRAMEWORK

Modeling framework description:

Hierarchical modeling favors the evaluation of various dependability metrics for a system by dividing the overall system model into different hierarchical layers consisting of sub-models. Hence, if the advantages of the state-space-based models are to model individual systems and focus on capturing operational states and state transitions of the system, the hierarchical models are used to model complex systems with the sophisticated architecture and hierarchy of different system/sub-system levels. In this paper, we propose a framework to model typical computer networks with three different sub-system levels. Each of the hierarchical levels captures the features of the corresponding level of the overall system. At the lowest level (leaf-level) we use SRN models to model every unit of the systems (e.g., the CPU, memory, power units, VMM, and VM etc. of a host). At the intermediate level (stem-level), fault-tree models are used to capture the architectural design of sub-systems (e.g., hosts and switches). Finally, at the highest level (root-level) of the hierarchical modeling framework, a RG is used to capture the network topology of the system. The hierarchical model is then described as an acyclic graph $\{\Psi, \Xi, \Gamma\}$, where, Ψ is the set of SRN sub-models $\psi_{srn}^{(2)}$ at the lowest level; Ξ is the set of links from the child sub-models $\psi_{srn}^{(2)}$ to the models at the higher corresponding level $\xi_{ft}^{(1)}$ at stem-level; and Γ is the set of the links from the child sub-models $\xi_{ft}^{(1)}$ of Ξ to the models at higher corresponding level $\gamma_{rg}^{(0)}$. In particular, the hierarchical model is formalized as follows:

- $\Psi := \{\psi_{srn}^{(2)}, \pi_{srn}^{(2)}\}$
- $\Xi := \{\psi_{srn}^{(2)}, \pi_{srn}^{(2)}, \xi_{ft}^{(1)}, \pi_{ft}^{(1)}\}$
- $\Gamma := \{\psi_{srn}^{(2)}, \pi_{srn}^{(2)}, \xi_{ft}^{(1)}, \pi_{ft}^{(1)}, \gamma_{rg}^{(0)}\}$

where, $\pi_{srn}^{(2)}$ is the output measure of the model $\psi_{srn}^{((2))}$ in the set Ψ, and is transferred to the higher-level model $\xi_{ft}^{(1)}$ in the set Ξ. In turn, $\pi_{ft}^{(1)}$ (the analysis result of the model $\xi_{ft}^{(1)}$) is forwarded to the higher-level model $\gamma_{rg}^{(0)}$. After all the analysis outputs of the lower-level models are transferred to the overall system model, the analyses of this highest-model enable the analysis results of the whole system to be taken in consideration.

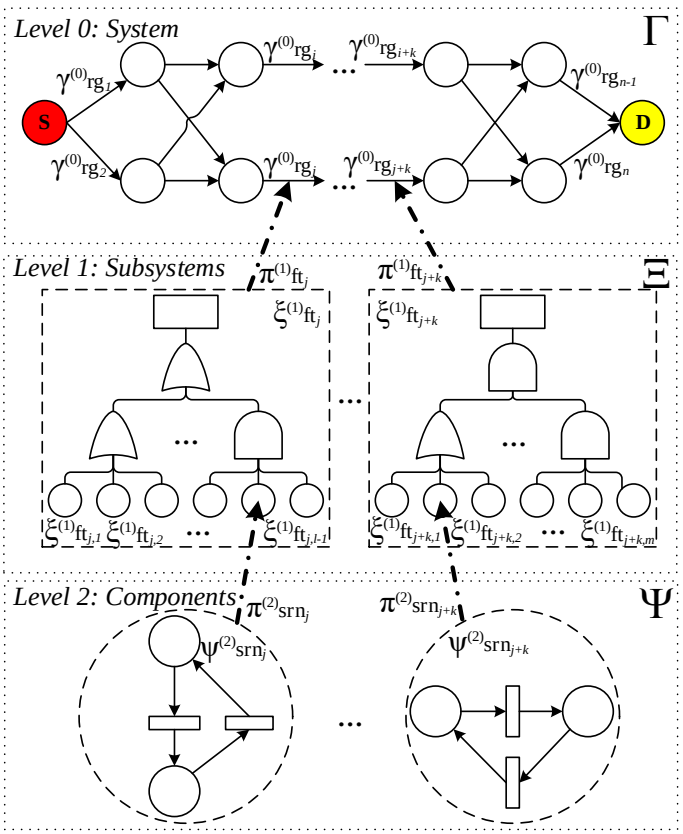


Figure 1: A hierarchical modeling framework for computer networks

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access

Fig. 1 depicts the entire set of processes of the proposed hierarchical modeling framework. At the lowest level (level 2 or leaf-level), the components of a computer network are modeled using SRN models $\psi^{(2)}_{srn_j}$ (which are hereby represented by places and timed transitions for simplicity) in order to capture the sophisticated operational states of the components in the most detailed manner as possible. After the set of these SRN models Ψ is solved and analyzed, the generated outputs are $\pi^{(2)}_{srn_j}$. In turn, these analysis results are used as the input parameters and are forwarded to the higher level models. At the intermediate level (level 1 or stem-level) the subsystems are modeled by FTs. Each FT represents for a respective physical subsystem in the considered system, for instance a server, or switch. Each leaf-event of the FT correspondingly represents a particular component of the subsystem. The input parameters of the leaf-events of the FTs $\xi^{(1)}_{ft_j}$ are replaced by the analysis results $\pi^{(2)}_{srn_j}$ of the corresponding SRN model $\psi^{(2)}_{srn_j}$. Subsequently, the analysis results of the FT $\xi^{(1)}_{ft_j}$ (which is $\pi^{(1)}_{ft_j}$) are then dispatched to the higher level models. All the FT models and their analysis results of the physical subsystems form the set Ψ at the middle level of the proposed hierarchical modeling framework. At the highest level (level 0 or root-level), the networking of the whole computer system is modeled using RG. Therein, the circle connecting the nodes only takes responsibility of intermediate place to attach the edges $\gamma^{(0)}_{rg_j}$ between the nodes. Each of the links $\gamma^{(0)}_{rg_j}$ represents for the modeling of a specific subsystem in the network. The analysis results $\pi^{(1)}_{ft_j}$ of the lower-level models $\xi^{(1)}_{ft_j}$ are used as the corresponding input parameters for the links $\gamma^{(0)}_{rg_j}$. The formation of all links $\gamma^{(0)}_{rg_j}$ which follows a predefined topology constitute the set Γ. In order to analyze the computer networks in accordance with a specific measure of interest, the models are in turn analyzed from the lowest level (the set of SRN Ψ models) to the intermediate level (the set of FT models Ξ) and eventually to the highest level (the overall RG model Γ). The analyses of all the models at different levels are conducted according to the same analytical measures of interest. The analysis outputs of the top-level model are the overall analysis results of the system in consideration.

A modeling example:

To facilitate comprehension of the above framework description, we shows the use of the framework for availability modeling of a specific computer network. We take one of the network configurations in [72] as shown in Fig. 2 as an example to demonstrate the construction of a hierarchical model for the system under consideration. The computer network consists of two physical servers (H1 and H2) connected to each other via a redundant network. The network devices include switches (SW1 and SW2) directly connected to the hosts, and routers (R1a and R1b on the H1 side, R2a and R2b on the H2 side). The two pairs of routers

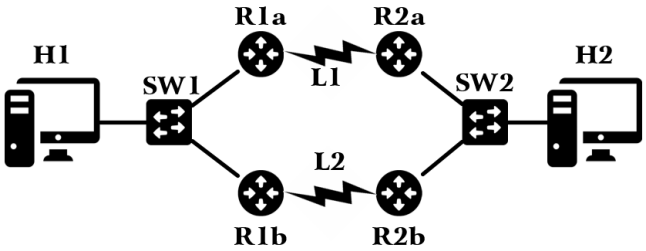


Figure 2: A typical computer network

(R1a and R2a), (R1b and R2b) constitute the networking via the corresponding links (L1 and L2). The redundancy of network devices and links aims to enhance the system's overall reliability and availability of the system. To evaluate this computer system with a simple network topology and fewer components, the authors in [72] used state-space models (CTMC and SPN models). We nevertheless demonstrate the construction of a hierarchical model for this computer system that complies with the proposed hierarchical modeling framework.

The construction of the hierarchical model for this computer network is shown in Fig. 3. The model comprises three levels from a top-down perspective (Γ, Ξ, Ψ) as proposed

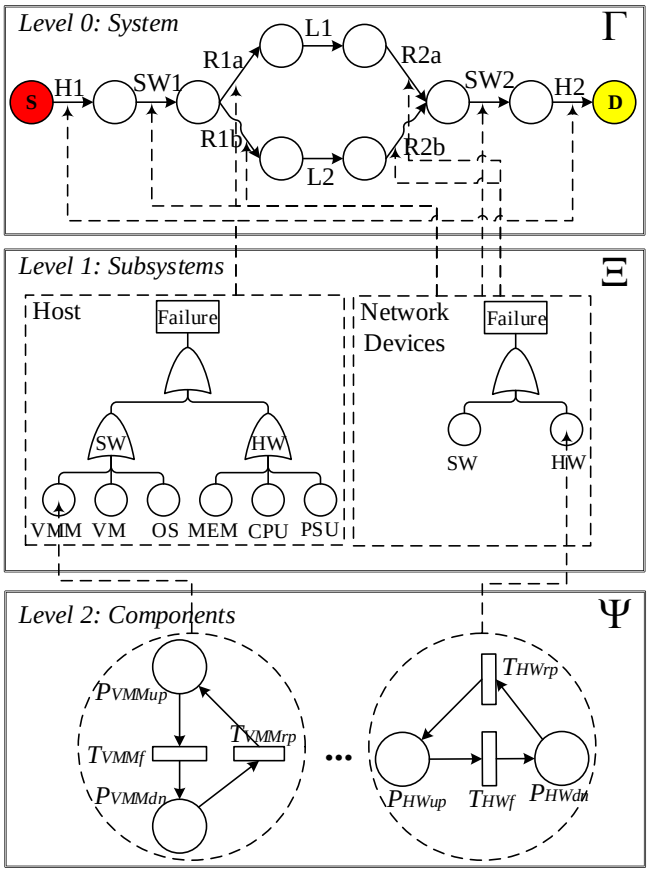


Figure 3: A hierarchical modeling model of a computer network

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access

in the hierarchical modeling framework. Supposed that we attempt to evaluate the availability of the network. The network is unavailable if there is no connection between the operation servers (H1 and H2). In other words, there is no continuous path from source node (S) throughout the RG at the level 0 (system model) of the hierarchical model to the sink node (D) as depicted in Fig. 3. Every arc between nodes in the RG system model represents for a subsystem which is in turn modeled in the lower subsystem models. The unavailability U of the network is logically formalized as in (1):

$$U = \overline{A} = \overline{A_{H1}} \vee \overline{A_{SW1}} \vee \left[\left(\overline{A_{R1a}} \vee \overline{A_{L1}} \vee \overline{A_{R2a}} \right) \wedge \left(\overline{A_{R1b}} \vee \overline{A_{L2}} \vee \overline{A_{R2b}} \right) \right] \vee \overline{A_{SW2}} \vee \overline{A_{H2}} \quad (1)$$

where, A_X and $\overline{A_X}$ represent, respectively the availability and unavailability of the corresponding component X in the network. Thus, the overall availability of the system is represented by a probability calculation as in (2).

$$\begin{aligned} A^p &= 1 - U^p \\ &= 1 - Pr\left\{ \overline{A_{H1}} \vee \overline{A_{SW1}} \vee \left[\left(\overline{A_{R1a}} \vee \overline{A_{L1}} \vee \overline{A_{R2a}} \right) \wedge \left(\overline{A_{R1b}} \vee \overline{A_{L2}} \vee \overline{A_{R2b}} \right) \right] \vee \overline{A_{SW2}} \vee \overline{A_{H2}} \right\} \\ &= 1 - Pr\left\{ \overline{A_{H1} \wedge A_{SW1} \wedge \left[\left(A_{R1a} \wedge A_{L1} \wedge A_{R2a} \right) \vee \left(A_{R1b} \wedge A_{L2} \wedge A_{R2b} \right) \right] \wedge A_{SW2} \wedge A_{H2}} \right\} \\ &= A^p_{H1} \times A^p_{SW1} \times \left[\left(A^p_{R1a} \times A^p_{L1} \times A^p_{R2a} \right) + \left(A^p_{R1b} \times A^p_{L2} \times A^p_{R2b} \right) \right] \times A^p_{SW2} \times A^p_{H2} \end{aligned} \quad (2)$$

where, A^p_X is the availability value of the corresponding component X in the network. Through this probability computation of the system availability, we also see that the redundancy of the networking explicitly enhances overall availability of the system. In our hierarchical model, the values of A^p_{H1} and A^p_{H2} are computed and substituted by the output generated from the FT model: Host in the lower level of subsystems. The values of A^p_{SW1}, A^p_{SW2}, A^p_{R1a}, A^p_{R1b}, A^p_{R2a}, and A^p_{R2b} (routers and switches) are then substituted by the analysis result of the same measure of interest from the FT model: network devices at the subsystems level. The values of A^p_{L1} and A^p_{L2}, which represent the availability of links between two consecutive components in the network are supposedly given in advance for this computer network.

In the stem-level (Ξ) of our hierarchical modeling framework, we use FT to capture the constitution of the architecture of every subsystem. In this example of the system, we suppose that a host comprises two main classes of components: (i) software components, which include the VMM, VM, and Operating System (OS); (ii) hardware components, which consist of the memory system (MEM), CPU and power supply unit (PSU). Each of the routers/switches is supposed for simplicity to consist of a hardware component (HW) and software component (SW). We assume that a certain failure in any of the above-mentioned components likely causes the total failure of the corresponding subsystem. Hence we use OR logical gates to connect the components in the FT models. The unavailability of a host $U_H = \overline{A_H}$ is logically formalized as in (3).

$$\begin{aligned} U_H = \overline{A_H} &= \overline{A_{H_{HW}}} \vee \overline{A_{H_{SW}}} \\ &= \overline{A_{VMM}} \vee \overline{A_{VM}} \vee \overline{A_{OS}} \vee \overline{A_{MEM}} \vee \overline{A_{CPU}} \vee \overline{A_{PSU}} \end{aligned} \quad (3)$$

In the probability calculation, we obtain the values of the availability of a host as in (4).

$$\begin{aligned} A^p_H &= 1 - U^p_H \\ &= 1 - Pr\left\{ \overline{A_{VMM}} \vee \overline{A_{VM}} \vee \overline{A_{OS}} \vee \overline{A_{MEM}} \vee \overline{A_{CPU}} \vee \overline{A_{PSU}} \right\} \\ &= 1 - Pr\left\{ \overline{A_{VMM} \wedge A_{VM} \wedge A_{OS} \wedge A_{MEM} \wedge A_{CPU} \wedge A_{PSU}} \right\} \\ &= Pr\left\{ A_{VMM} \wedge A_{VM} \wedge A_{OS} \wedge A_{MEM} \wedge A_{CPU} \wedge A_{PSU} \right\} \\ &= A^p_{VMM} \times A^p_{VM} \times A^p_{OS} \times A^p_{MEM} \times A^p_{CPU} \times A^p_{PSU} \end{aligned} \quad (4)$$

In a similar manner, we can formalize and compute the unavailability and availability of network devices (routers and switches), respectively, as in (5) and (6).

$$U_{ND} = \overline{A_{ND}} = \overline{A_{HW}} \vee \overline{A_{SW}} \quad (5)$$

$$\begin{aligned} A^p_{ND} &= 1 - U^p_{ND} \\ &= 1 - Pr\left\{ \overline{A_{HW}} \vee \overline{A_{SW}} \right\} \\ &= 1 - Pr\left\{ \overline{A_{HW} \wedge A_{SW}} \right\} = Pr\left\{ A_{HW} \wedge A_{SW} \right\} \\ &= A^p_{HW} \times A^p_{SW} \end{aligned} \quad (6)$$

where, $\overline{A_X}, A_X, A^p_X$, respectively, represent unavailability, availability and its value of the corresponding component X in the subsystems. The values of availability $A^p_{VMM}, A^p_{VM}, A^p_{OS}, A^p_{MEM}, A^p_{CPU}, A^p_{PSU}$ (for a host) and A^p_{HW}, A^p_{SW} (for a network device) are computed and substituted by the analysis outputs of the corresponding component models in the lowest level (Ψ) of the proposed hierarchical modeling framework.

In the leaf-level of component (Ψ), a number of SRN models are constructed to capture the operational states and state transitions of every components in a host or a network device. To simplify the computation of the measures of interest for the system of the example, we use a two-state SRN modeling approach in which the operational state and unavailable state of each component are captured by the respective UP and DOWN states in the SRN model. For instance, as shown in Fig. 3, the places

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access

P_{VMMup} and P_{VMMdn} represent the UP and DOWN states of a VMM in a certain host. The transitions in the SRN model of a VMM (T_{VMMf} and T_{VMMrp}) are to capture the failure and repair behaviors of the VMM. The same construction is applied for the remaining components. The places P_{HWup}, P_{HWdn} depict the UP and DOWN states of the hardware subsystem in a network device, whereas the transitions T_{HWf}, T_{HWrp} represent the failure and recovery of that hardware component. A specific measure of interest is computed by defining reward functions by assigning appropriate reward rates to the states of SRN models. The states in which the component is functioning are associated with a reward rate of 1. A reward rate of 0 is assigned to the failure states. We present two reward functions for the two components, VMM in a host and HW in a network device respectively as in (7) and (8) .

$$r^{VMM} = \begin{cases} 1 : if\#(P_{VMMup}) == 1 \\ 0 : otherwise \end{cases} \tag{7}$$

$$r^{HW} = \begin{cases} 1 : if\#(P_{HWup}) == 1 \\ 0 : otherwise \end{cases} \tag{8}$$

Here, r^{VMM}, r^{HW} respectively, represent the reward rates assigned to the states (UP or DN) in the VMM and HW models. The symbol $\#$ indicates the number of tokens in the corresponding places. The reward functions for other component SRN models are inferred in a similar manner.

The availabilities (the measure of interest in consideration) of the components are then computed as the expected reward rate $E[X]$ and expressed as in (9):

$$E[X] = \sum_{j \in \Pi} r_j . \pi_j \tag{9}$$

where, X is the random variable that represents the steady-state reward rate of a measure of interest, Π is the set of tangible markings of the corresponding SRN model, π_j is the steady-state probability of the marking j, and r_j is the reward rate in the marking j as defined in the above-mentioned reward functions.

The formal representation of the hierarchical model for the system in consideration is given as follows:

$$\begin{aligned}
\Psi :=& \Big\{ (\psi_{srn}^{VMM}, \pi_{srn}^{VMM}); (\psi_{srn}^{VM}, \pi_{srn}^{VM}); (\psi_{srn}^{OS}, \pi_{srn}^{OS}); \\
& (\psi_{srn}^{MEM}, \pi_{srn}^{MEM}); (\psi_{srn}^{CPU}, \pi_{srn}^{CPU}); (\psi_{srn}^{PSU}, \pi_{srn}^{PSU}); \\
& (\psi_{srn}^{HW}, \pi_{srn}^{HW}); (\psi_{srn}^{SW}, \pi_{srn}^{SW}) \Big\} \\
\Xi :=& \Big\{ \Big[(\psi_{srn}^{VMM}, \pi_{srn}^{VMM}), \xi_{ft}^{H}, A_{VMM}^{P} \Big]; \Big[(\psi_{srn}^{VM}, \pi_{srn}^{VM}), \\
& \xi_{ft}^{H}, A_{VM}^{P} \Big]; \Big[(\psi_{srn}^{OS}, \pi_{srn}^{OS}), \xi_{ft}^{H}, A_{OS}^{P} \Big]; \Big[(\psi_{srn}^{MEM}, \pi_{srn}^{MEM}), \\
& \xi_{ft}^{H}, A_{MEM}^{P} \Big]; \Big[(\psi_{srn}^{CPU}, \pi_{srn}^{CPU}), \xi_{ft}^{H}, A_{CPU}^{P} \Big]; \\
& \Big[(\psi_{srn}^{PSU}, \pi_{srn}^{PSU}), \xi_{ft}^{H}, A_{PSU}^{P} \Big] \Big\} \\
\Gamma :=& \Big\{ \psi_{srn}^{(2)}, \pi_{srn}^{(2)}, \xi_{ft}^{(1)}, \pi_{ft}^{(1)}, \gamma_{rg}^{(0)} \Big\}
\end{aligned} \tag{10}$$

The above hierarchical models are developed and aggregated in the symbolic hierarchical automated reliability and performance evaluator (SHARPE)[73] . SHARPE allows the use of eight different types of models which are in turn aggregated to construct a sophisticated hierarchical model of a complex system in practice [73, 74]. We will use the proposed hierarchical modeling framework to evaluate reliability/availability of a typical DCN based on a fat-tree network topology.

IV. DATA CENTER NETWORKS

In this section, we describe two representative DCNs based on three-tier and fat-tree topologies which are widely used in industry, which are to be compresensively modeled and studied using the above-proposed modeling framework. Fig. 4b. Small-sized three-tier and fat-tree based DCNs are shown in Fig. 4a and Fig. 4b. The DCNs commonly consist of 16 physical servers connected to each other via a three-tier/fat-tree based network topology consisting of three levels of switches. In the three-tier DCN, the top level is comprised of core switches ($C_i, i = 1, ..., 2$) which are to connect the DCN to outer environment. The core switches are connected to each other and divide the network in branches respectively to each core switch. The middle level consists of aggregation switches ($A_j, j = 1, ..., 4$). The aggregation switches are linked to all the core switches. The bottom level consists of access switches ($E_k, k = 1, ..., 8$) which are all connected to each pair of aggregation switches and sixteen physical servers ($H_l, l = 1, ..., 16$). Whereas, the fat-tree based network topology is also comprised of three layers of switches. The top layer consists of core switches ($C_i, i = 1, ..., 4$) to link different pods to each other. The middle layer consists of aggregation switches ($A_j, j = 1, ..., 8$) to integrate and direct the data traffic within a pod. The lower layer consists of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access

edge switches ($E_k, k = 1, ..., 8$) to connect sixteen physical servers ($H_l, l = 1, ..., 16$) with the upper switches. The three-tier DCN has a less number of but more costly switches and more high-speed links in comparison to the fat-tree DCN. We consider a common practical case in high-performance and HA computing systems in DCs for both DCNs to be modeled. That is, the continuous data connection and transactions between compute nodes in a computer network requires high reliability/availability in parallel computing problems. In order to prolong the connection between the compute nodes, selecting an appropriate routing topology at a given time to avoid component failures will improve the availability/relaibility of the connection. Without loss of generality, in this paper, we investigate the data connection and transactions between four fixed compute nodes (H1-H4) and four other non-fixed compute nodes in the network. These non-fixed nodes are selected so that the highest availability/reliability can be achieved. Fig. 5 and Fig. 6 show six case-studies of the three-tier and fat-tree DCNs , respectively. As we examine in detail, without loss of generality, these six cases has covered all the different cases of the connection between a cluster of fixed comptute nodes with four other non-fixed compute nodes in the network. For instance, when considering the case I of three-tier DCN (Fig. 5a), it is the unique routing topology between the four compute nodes (H1-H4) with the four other active nodes (H5-H10). Whereas, in the case II (Fig. 5b), if we select the pair of nodes (H7-H8) as active nodes instead of (H5-H6), we obtain another same variant with the one shown in the subfigure. If we assign the pair of nodes (H11-H12) (or other pairs) to be active nodes instead of (H9-H10), we also obtain another same variant. Therefore, the routing topology shown in Fig. 5b is a unique representative of all variants in the case II. The above inference is applied in the similar manner for other cases of both DCNs. We investigate the impact on reliability/availability of the distribution of the non-fixed compute nodes in the network by moving the nodes within and between pods. In DCNs, the network controller may change the routing path over time. However, it is necessary to guide the network controller to choose a destination among compute nodes to secure highly continuous network connectivity and to enhance reliability/availability of the network. The importance of reliability/availability in a DC was previously demonstrated in [75]. These researchers also mentioned that, the reliability/availability of a DC is basically composed of not only the reliability/availability of computing servers but also the reliability/availability of network connections. In this paper, we focus on the network topologies and connections of servers. The higher reliability/availability the system can achieve, the higher capability of connection the system can obtain.

Several modeling formalisms are commonly used for reliability/availability quantification of a specific system. The most suitable formalism for each specific case of a practical system is selected often in the consideration of the following important factors as presented in [47]:

- the system's architecture and featured behaviors to be considered
- the chosen modeling formalism's modeling power and tractable representation of the system's features and properties
- the reliability/availability attributes of interest

In accordance with the above factors, ones may consider the following classification of model types:

- *Monolithic* models developed often by a single formalism
- *Multi-level* models often used for modeling multi-level complex systems by *a divide and conquer strategy*.

Modeling power of multi-level models:

It is likely infeasible for a single formalism to capture the complexity and sophisticated behaviors of real-life systems in an adequate manner. If non-state-space-based approaches have limited capability, state-space-based formalisms are highly capable to incorporate interdependencies and dynamic transitions in system behaviors. [47] Nevertheless, the latter often suffer from largeness and state-space explosion problems. In this study, the combining three different formalisms is proposed to develop a multi-level model. Suitable formalisms are selected to represent behaviors and architectures of component or subsystem levels, and then the submodel results are composed and passed upto system model in order to obtain overall measures of interest. We will apply a multi-level modeling framework as previously proposed to quantify reliability/availability attributes of tree-based DCN complying with real-world network topologies as described in this section.

Largeness tolerance and avoidance of multi-level models:

Reliability and availability quantification of sophisticated systems using modeling formalisms often suffers from largeness and stiffness problems [47]. Two approaches to confront with the problems are avoidance and tolerance techniques. Non-state-space models including RGs, and FTs in multi-level models are usually considered as largeness avoidance modeling techniques. In these approaches, smaller models are generated and then, the solutions of such models are combined and rolled up to come up with the overall model solution. This way is often feasible to seperate the system model into different levels (which also requires the dividing of system architecture into multi-levels), thus help avoid and tolerate the largeness/stiffness problems in modeling.

Three-level network architecture:

Our focus of interest is on a network of systems complying with a tree-based topology which is in practice a multi-level complex system. The system architecture itself has a nature of three levels consisting of:

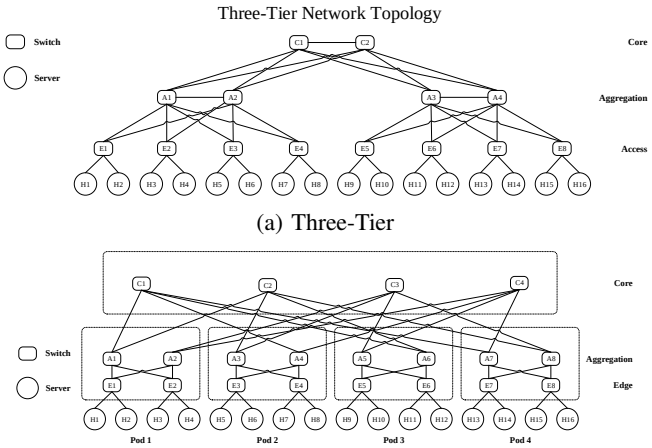
- *Top level or system level* is characterized by a specific network topology (here, tree-based topology). The routing at a time is supposed to be software defined

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access

which is to say that the routing can be specified in a flexible manner to obtain the higher reliable data transactions within the network topology.

- *Middle level or subsystem level* consists of heterogeneous physical subsystems including network devices (switches) and servers.
- *Ground level or component level* consists of components which are the smallest divisions of physical subsystems in the middle level.

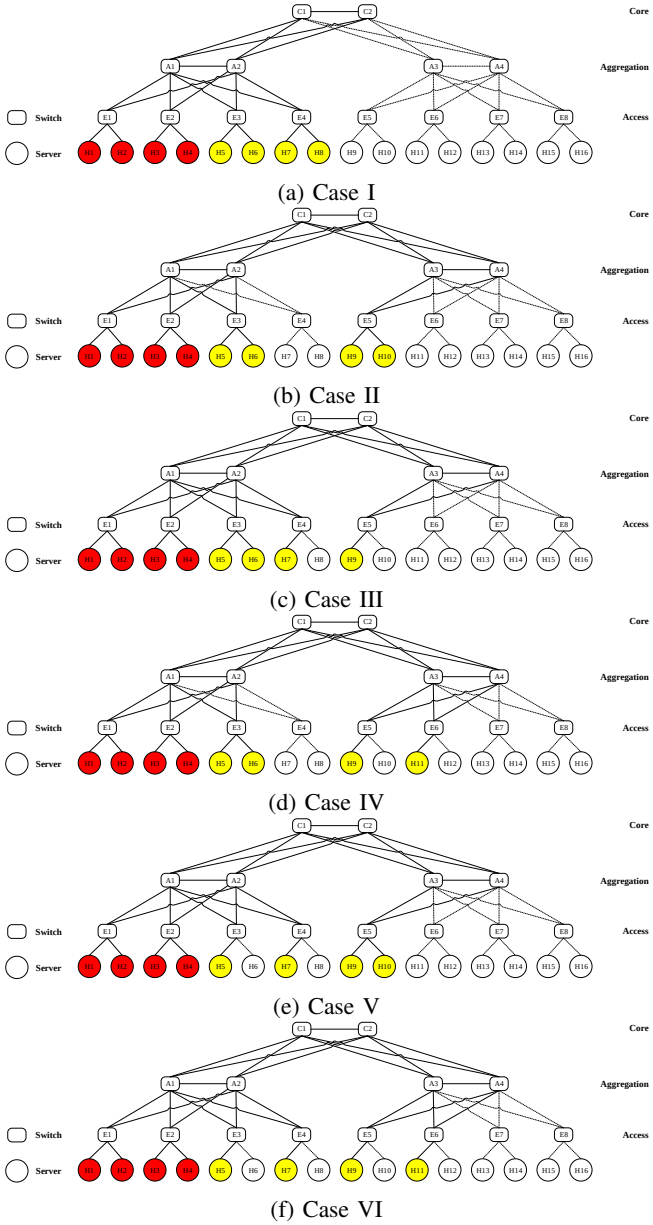
The dividing of the system into three levels actually helps us manage to develop suitable models for each level. In particular, RG models can capture routings in the topology in the top level, while FT can represent underlying compositions of failure and repair of a physical subsystems in the middle level, and SRN can comprehensively capture detailed operation within a component in the ground level. The use of multi-level modeling in a hierarchical manner with different formalisms in each level can ease the modeling while comprehensively characterizing and capturing featured system structure and behaviors.



(a) Three-Tier

(b) Fat-Tree

Figure 4: DCN Topologies



(a) Case I

(b) Case II

(c) Case III

(d) Case IV

(e) Case V

(f) Case VI

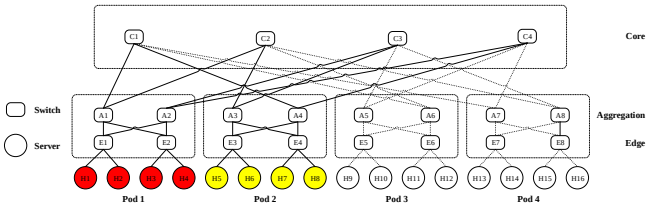
Figure 5: Case-studies of three-tier DCN

V. HIERARCHICAL MODELS

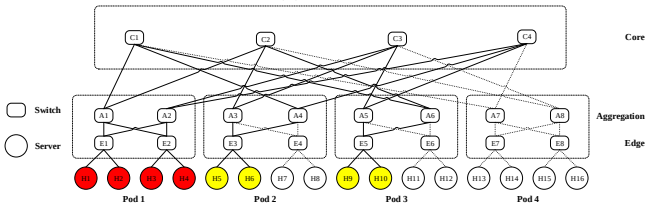
In this section, we will describe the detail of hierarchical models in top-down perspectives for the DCNs in consideration. At first, we present several modeling assumptions as follows.

A model is often a mathematical abstraction of the system aimed at providing a simplified conceptualized representation of the behavior of the system and thus preserve the main features of the system [47]. In this study, the modeling framework is devoted to capture the complexity and flexibility of network topologies at the network level, and to take into account failure and recovery behaviors of individual network elements at the subsystem level, and at last to incorporate sophisticated state transitions within components of each above network element at the subsystem level. The modeling aims to capture a variety of heterogenous features of the system from the highest level of network topologies down to the lowest level of state transitions of components/units while harmonizing the complexity and largeness level of overall system model. The framework is literally usable for system assessment under a variation of network topologies and a broad range of system architectures and underlying operational behaviors within components acting as building blocks of the system. Therefore, in accordance with system design requirements, ones may take into account different system behaviors and architectures in the modeling. Nevertheless, if we incorporate sophisticated interactions between subsystems, we may confront with largeness problem and also we may neglect the efforts of capturing overall network topologies as well as the

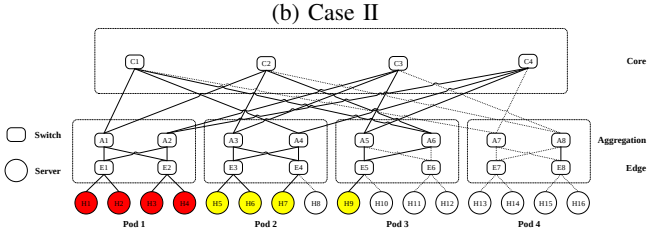
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access



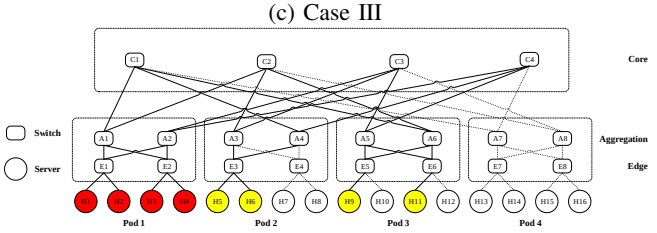
(a) Case I



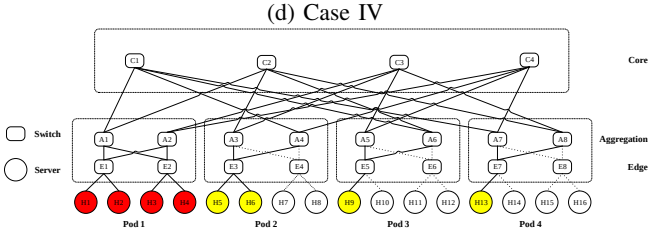
(b) Case II



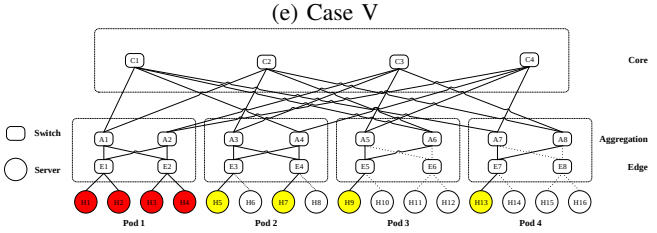
(c) Case III



(d) Case IV



(e) Case V



(f) Case VI

Figure 6: Case-studies of Fat-Tree DCN

architectures of individual network elements, and vice versa. For this reason, it is necessary to make some assumptions in modeling due to several drawbacks of non-state space formalisms including RG and FT as well as of state-space formalism like SRN in the proposed framework, as follows:

- We assume that the system is under a strict requirement of system design in which a failure causing an outage of any component consituting a network element (servers and switches) is considered leading to a total failure of that element in the network. In particular, we assume on a system user's perspective that the whole number of VMs running on a host represents and consitutes an individual component of VMs of the host regardless the origin of each individual running VM on the host. The whole number of VM is, thus assumed to be an average number of VMs hosted on the physical server. The availability of the VMs in a host is strictly defined as at least a VM is in normal state P_{VMup} or failure-probable state P_{VMfp}.
- To relax the complexity of modeling in order to focus on the overall archiecture of the network, it is necessary to assume that sophisticated interactions and dependences among network elements, for instance those of VM migration, are represented roughly by the connectivity of data transactions within the network and thus, are not taken into account in detail in the modeling due to several modeling limitations of non-state space formalims like RG and FT. Further incorporation of such interactions can be achieved by a monolithic or interacting state-space model as shown in some of previous works [35, 65, 76–78] with a sacrifice of not considering the overall architecture of network topologies and network elements.
- Basic events of FTs in practice can actually be either a binary basic event (which can be found in one of two mutually exclusive and exhaustive states (UP or DOWN)), or a multi-state basic event (which is represented by more than two states) [47]. For this reason, we assume to use basic events to represent links/connections between network element (which is captured by mean time to failure (MTTF) λ_L and mean time to repair (MTTR) μ_L of links), whereas we focus on exploring multi-state operations and underlying architecture of network elements including physical servers and switches.
- FTs of real-world systems are probably sophisticated as shown in [31] in which the FTs can be either coherent FT or non-coherent FT consisting of different gates such as AND gate, NOT gate and derived gates like XOR. And the state-space sub-models of basic events can also incorporate a variety of practical behaviors in real-world circunstances such as imperfect repair processes, the involvement of a pool of repair persons or a single, shared repair person among the subsystems and components etc. These sophisticated operations can be incorporated in the state-space models and/or FTs. Nevertheless, we limit ourself to simplify the FT models and underlying state-space models. The proposed framework is helpful for practitioners to apply in real-world systems and relevant circumstances.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access

A. SYSTEM MODELS

The RG system models of three-tier and fat-tree DCNs as shown respectively in Fig. 7 and Fig. 8 are developed in accordance with the corresponding routing of the case-studies presented in Fig. 5 and Fig. 6.

As described in the hierarchical modeling framework, each RG is made up of edges and circles. The edges are used to denote system components such as hosts, switches (with the same symbols as in case-studies in Fig. 5 and Fig. 6) and links between the components. The circles are used as connecting points of the edges. There are two special nodes, S and D, which are the start and end node of RG, respectively. Stochastic metrics of interest of system components are calculated using lower level models and are attached to edges as input distribution functions (as described in the hierarchical modeling framework). We denote the edges representing the links between the two parts of the system in words which are combined by the notation of the two parts. For example, the link between host H1 and switch E1 is the H1E1 edge. The same notations are applied to other links in the system. These RGs also show that routing topologies and connectivity between four fixed compute nodes in the network with four non-fixed compute nodes are captured by continuous paths from node S to node D consecutively interconnecting edges and circles with each other. Thus, the reliability/availability of the network depend on the reliability/availability of each component in the network and at the same time, on the predetermined routing topologies. The network is considered as reliable/available if there is a consecutive path between the nodes S and D. More specifically, the reliability/availability of the DCN is indicated by the continuity of all possible connections between the two nodes S and D. The network's reliability/availability is evaluated after calculating the reliability/availability of its subsystems and components that are represented by the edges in RGs. In the following sections, we will detail the development of stochastic models for subsystems and components of the DCN.

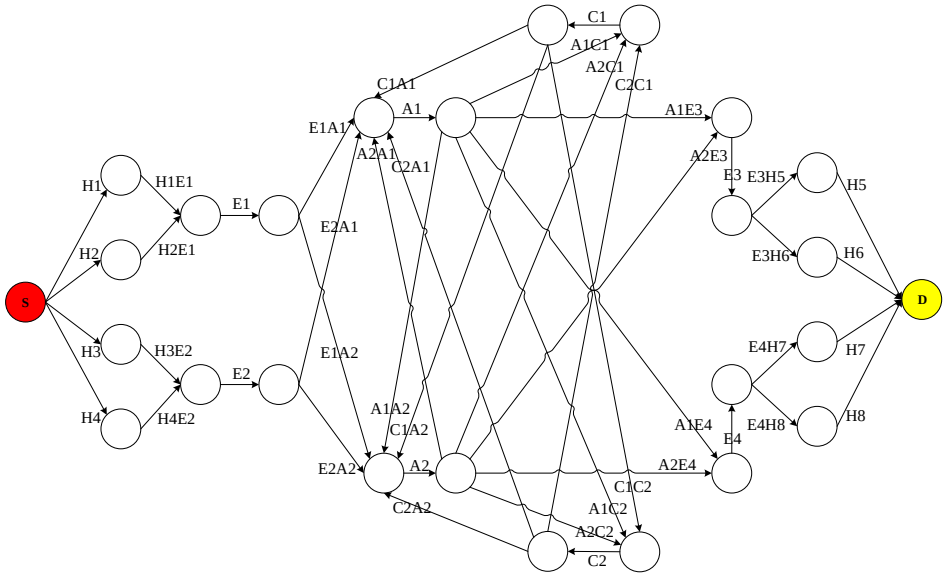
B. SUBSYSTEM MODELS

This section describes the modeling of subsystems and components in detail.

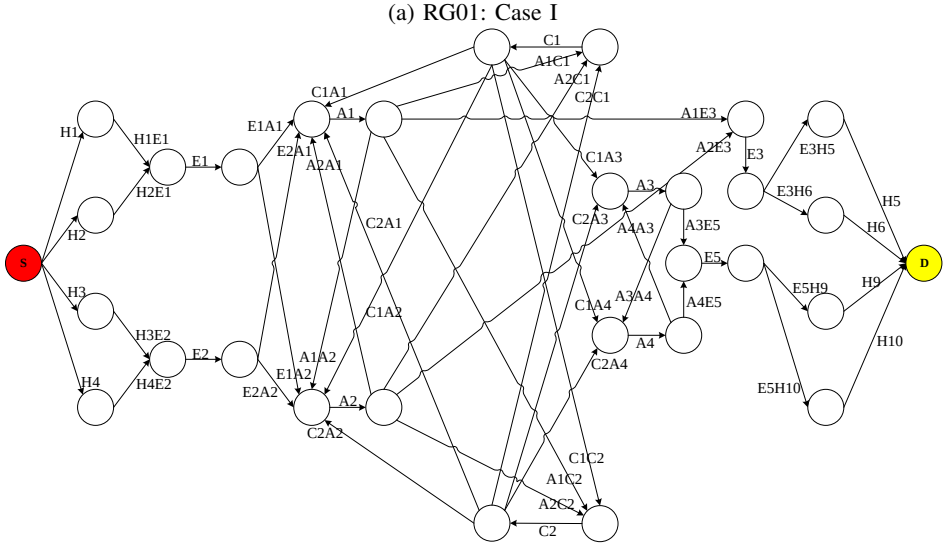
1) Modeling of a host

i. ***System fault-tree model of a host:***
The models of a host are shown in Fig. 9. The failure/unavailability of the host is captured using a fault-tree as in Fig. 9a. A host becomes unavailable if either the hardware (HW) or Software (SW) subsystems fail. Thus, the branches HW and SW in the fault-tree represent the hardware and software subsystems of the host. The leaf nodes in the fault-tree correspond to every subsystems at device level. Respectively, HW consists of the center processing unit (*CPU*), memory (*MEM*), network (*NET*), power (*PWR*), and cooler (*COO*). Further, SW consists of VMM (*VMM*), VM (*VM*) and applications (*APP*). The leaf nodes are further incorporated of the corresponding SRN models of the subsystems at the lower level in the hierarchy as respectively depicted in the Fig. 9b to Fig. 9g.

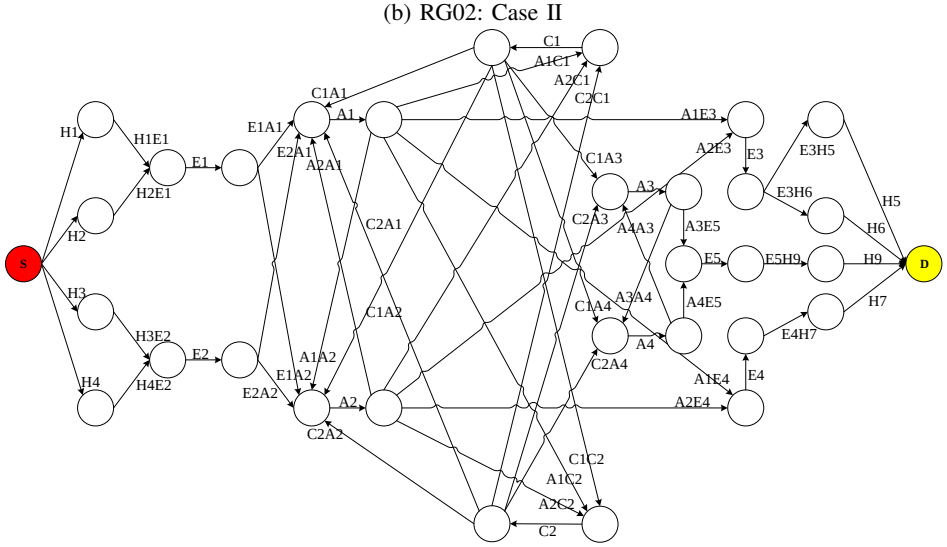
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access



(a) RG01: Case I

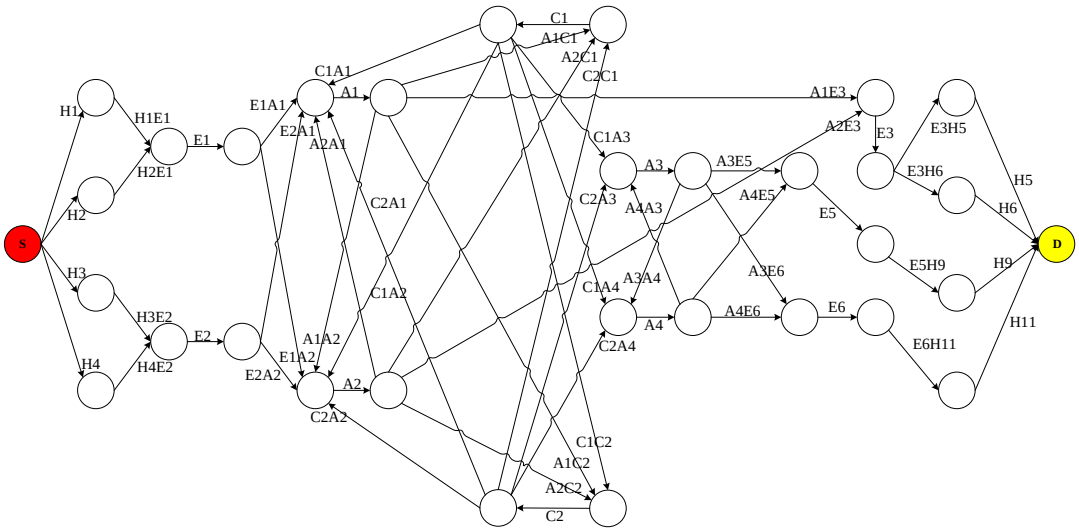


(b) RG02: Case II

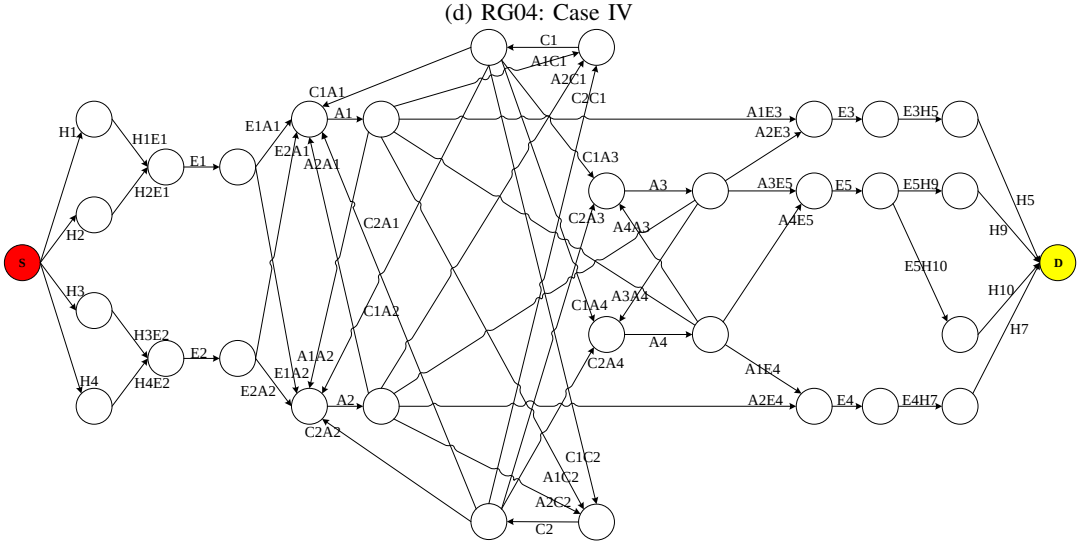


(c) RG03: Case III

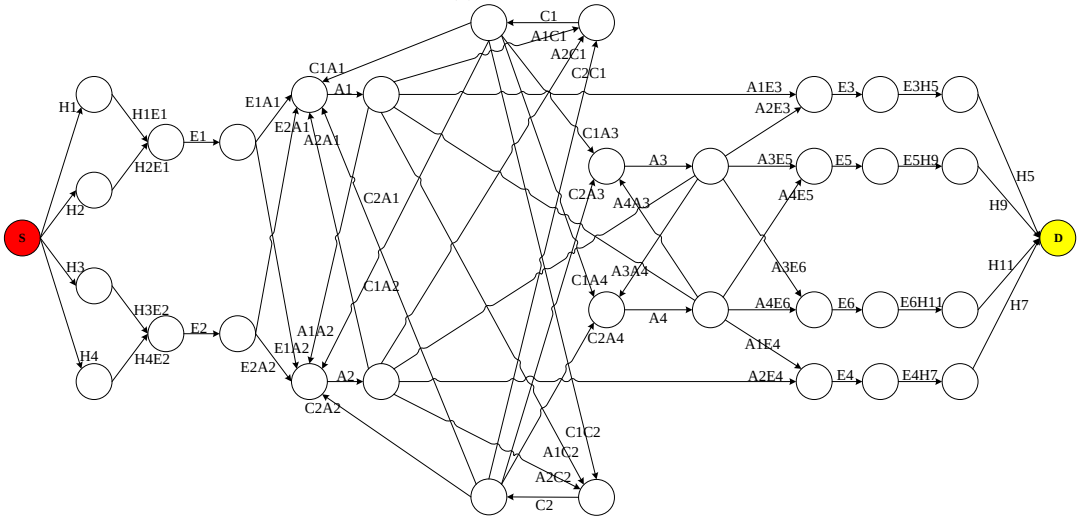
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access



(d) RG04: Case IV



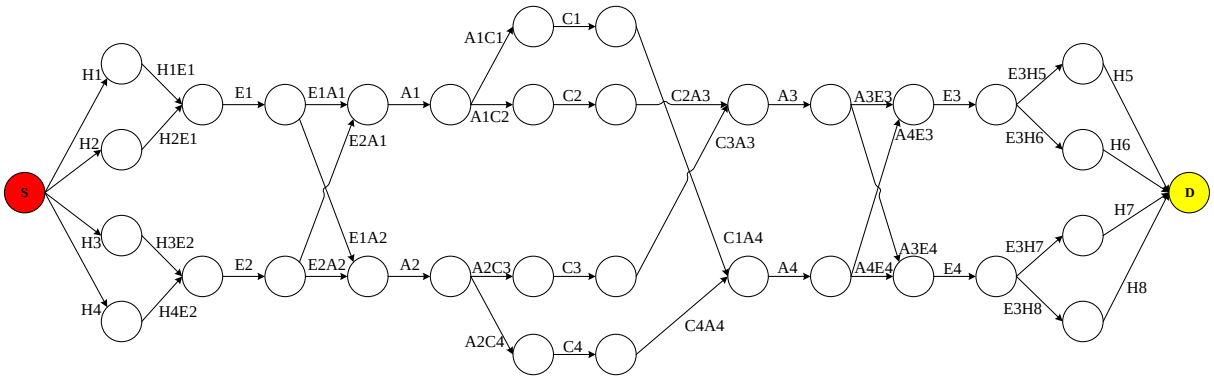
(e) RG05: Case V



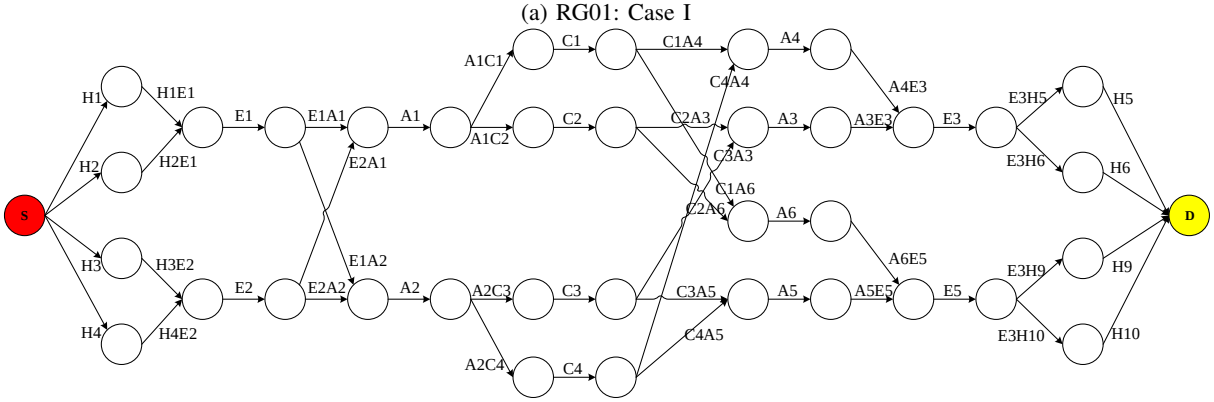
(f) RG06: Case VI

Figure 7: Reliability Graphs of a Three-Tier Data Center Network

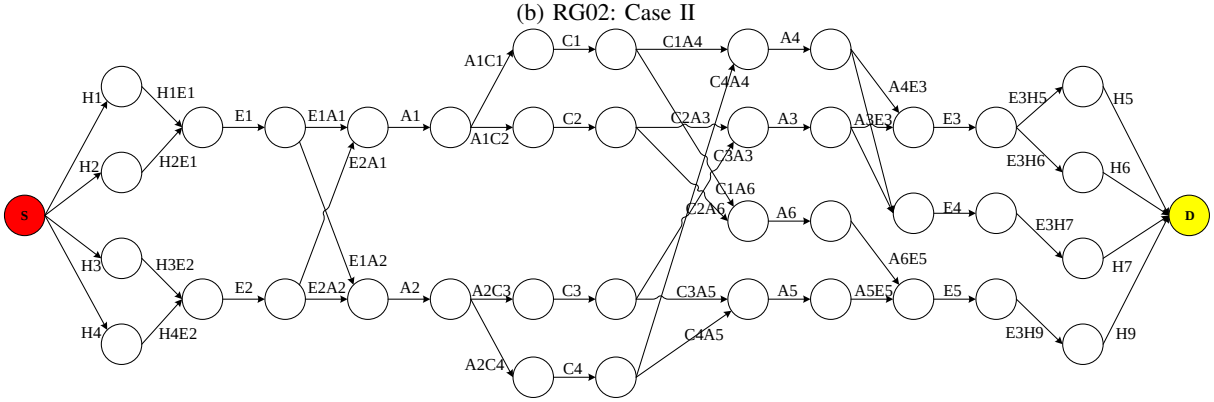
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access



(a) RG01: Case I

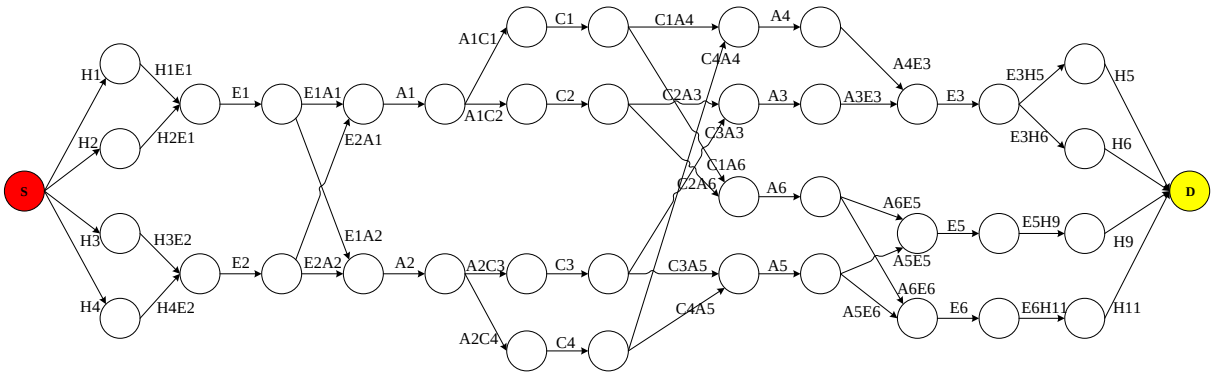


(b) RG02: Case II

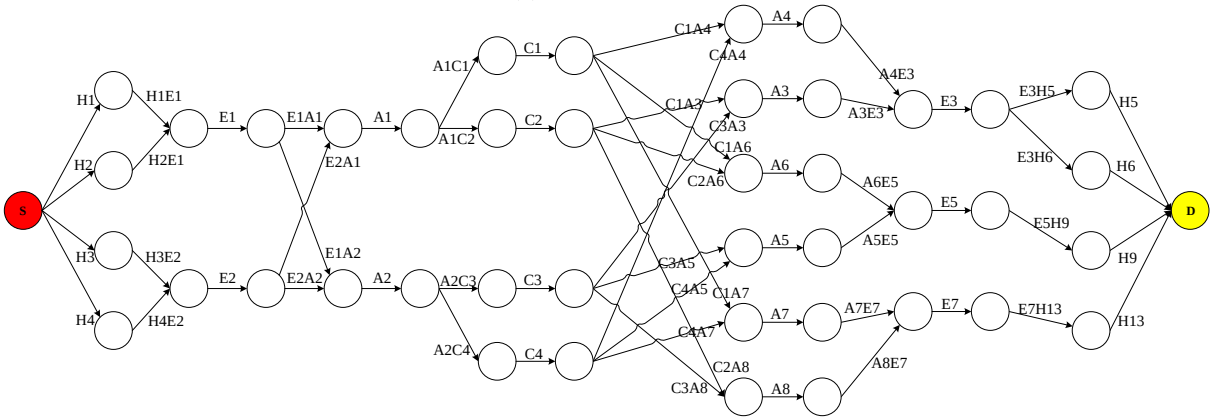


(c) RG03: Case III

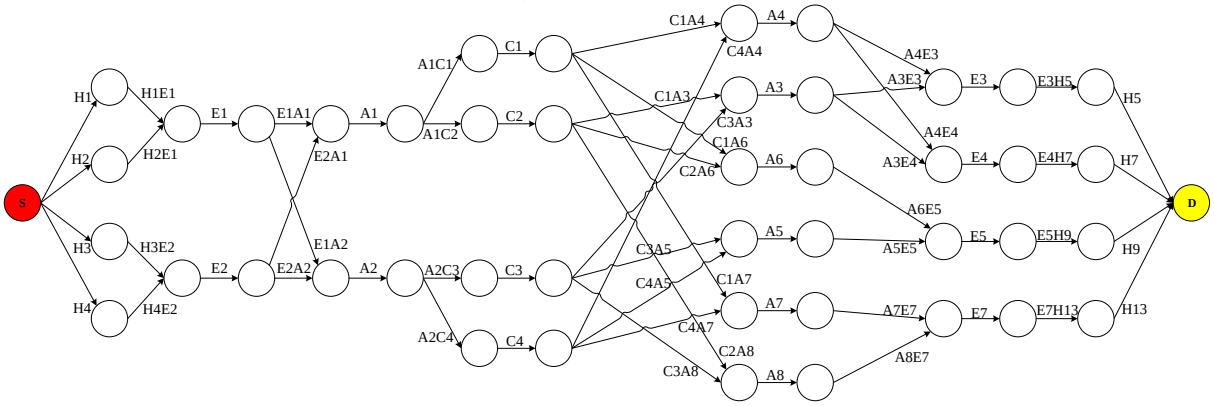
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access



(d) RG04: Case IV



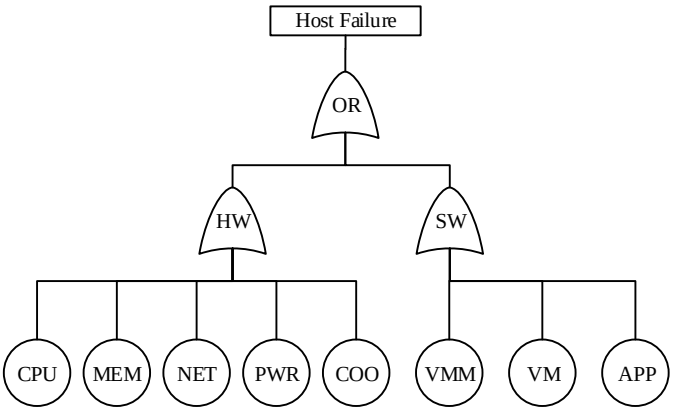
(e) RG05: Case V



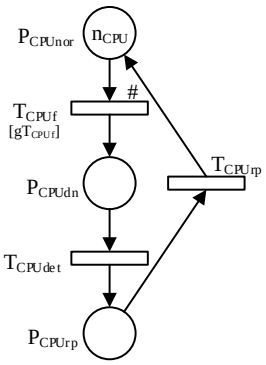
(f) RG06: Case VI

Figure 8: Reliability Graphs of a Fat-Tree Data Center Network

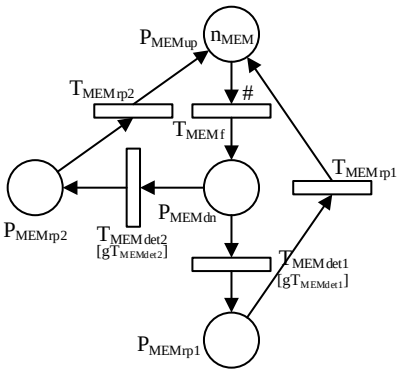
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access



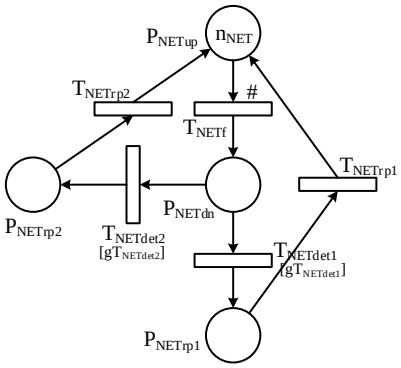
(a) Fault Tree of a Host



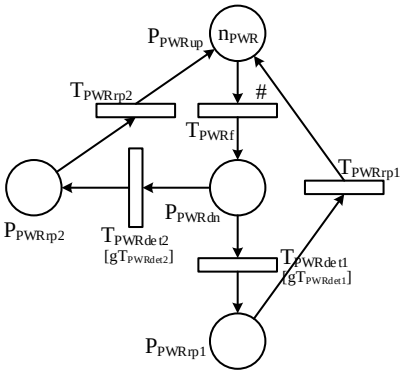
(b) CPU



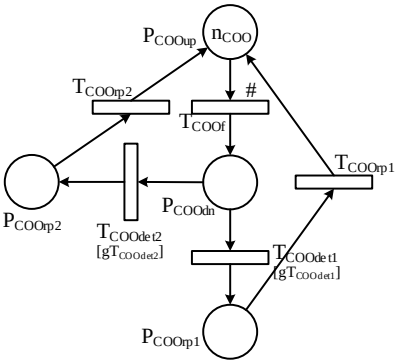
(c) MEM



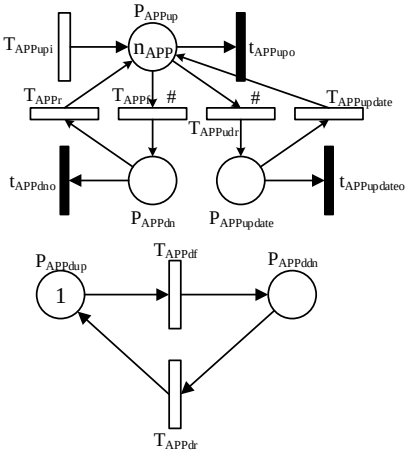
(d) NET



(e) PWR



(f) COO



(g) APP

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access

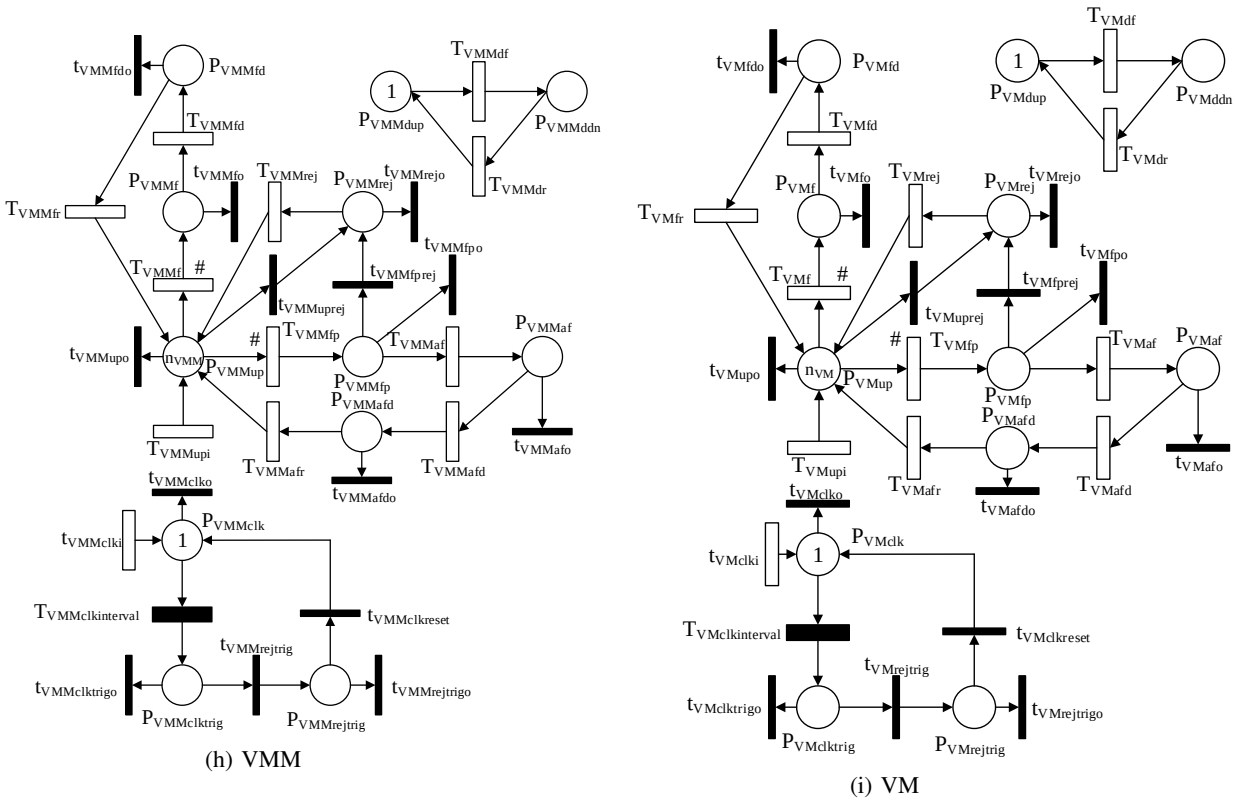


Figure 9: Sub-models of a Host

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access

ii. ***Subsystem level models of a host:***

SRN model of CPU subsystem

Fig. 9b depicts the SRN model of the CPU subsystem. As supposed, a host may comprise of n_{CPU} number of CPUs running in normal-state at the beginning, represented as the n_{CPU} tokens in the place P_{CPUnor}. As time proceeds, a CPU fails and enters the downstate. Since one of the CPUs in the processing system fails, the remaining CPUs stop functioning. The failure of a CPU is represented by the firing of the transition T_{CPUf} with the firing rate of $\#(P_{CPUnor}).\lambda_{CPU}$ (marking dependence depicted by the $\sharp$ sign). Subsequently, one token in the place P_{CPUnor} is removed and deposited in P_{CPUdn}. When a CPU (as a part of the hardware system) goes down, the CPU subsystem and consequently, the whole server also fails. Thus the condition under which the CPU functions normally is when the full number of CPUs, n_{CPU}, are in the running state. The transition T_{CPUf} halts the transition of tokens. As soon as a CPU enters downtime (a token is deposited in P_{CPUdn}), a repairperson is summoned to detect the faults and then repair or replace the failed CPU. The transition T_{CPUdet} fires to represent the detection of the repairperson. The token in P_{CPUdn} is removed and deposited in P_{CPUrp} for the repair. The transition T_{CPUrp} depicts the repair of the CPU. After the repair, the transition T_{CPUrp} fires and the token in P_{CPUrp} is removed and deposited in P_{CPUnor}. The number of tokens in P_{CPUnor} satisfies the condition for the CPU subsystem to function normally.

The SRN models of the other subsystems are shown in Fig. 9c to Fig. 9f, respectively, including the memory (Fig. 9c), network (Fig. 9d), power supply (Fig. 9e) and cooler (Fig. 9f). Although these subsystems control different functionalities, we observe and take into account their similar failure modes and recovery behaviors. The SRN models therefore have the same pattern consisting of similar places, transitions and model behaviors. We describe the SRN model for the memory subsystem and the description of other models is referred accordingly.

SRN model of MEM subsystem

The real-world memory systems could be more sophisticated and the operational states of memory systems in real-world server systems could be more complicated depending on the internal architectures as shown in many of previous works: [31, 79, 80]. However, in this study, we limit ourself to consider specific cases in maintainence for a distributed memory system without loss of generality which are popular in data center systems as follows:

- As any of the running memory elements fails due to uncertain faults, a repair person is summoned to diagnose/repair/replace the failed element. There could be a certain number of memory elements in the failure state but the overall memory system can still provide enough space and services for normal operations of the upper host. In this case, the repair-person arrives to diagnose/repair/replace in advance of a threshold of the failed memory elements to consider the memory system in a failure state. After the maintenance, the memory system is in its normal state.
- In the case that the repair person does not arrive in time, the number of failed memory elements is over the threshold causing an onverall failure of the memory system. For this reason, the processes of diagnosis and recovery would obviously take longer time compared to those in the above case. The repair person has to possibly diagnose the whole memory system, reinstall and restart the host system.

The MEM subsystem undergoes four main states in modeling as shown in Fig. 9c including: P_{MEMup} (memory components are in normal operation); P_{MEMdn}(one of the memory components fails because of a certain failure); P_{MEMrp1} (the failed memory component is being repaired by a summoned repairperson in case the memory subsystem is still running even though some of the memory components failed); and P_{MEMrp2} (the memory components are repaired after the whole memory subsystem failed to provide sufficient services). The system initiates with n_{MEM} in running state P_{MEMup}. After a period of operation, one of the memory components fails and enters downtime state P_{MEMdn}. This failure is captured by the transition T_{MEMf}. After the firing of this transition, one token is removed from P_{MEMup} and deposited in P_{MEMdn}. Since a number of memories are in upstate P_{MEMup} simultaneously, they compete to fail and thus the transition rate of T_{MEMf} depends on the number of remaining tokens in P_{MEMup} (which is known as marking dependence). As soon as a memory component fails (in which the memory subsystem does not provide sufficient memory capacity), a repairperson is summoned to detect and repair/reinstall a new memory device. The detection and repair operations are captured by the transitions $T_{MEMdet1}$ and T_{MEMrp1}. As soon as the repairperson is summoned and the detection is completed, the transition $T_{MEMdet1}$ is fired, then one token in P_{MEMdn} is removed and deposited in P_{MEMrp1}. When the repair or new installation is completed, the transition T_{MEMrp1} is fired and the aforementioned token is removed and deposited in P_{MEMup}. In another case, when a number of memories remain in the downtime period (same number of tokens reside in the place P_{MEMdn}) but the repairperson does not arrive in time which causes the overall failure of the memory subsystem (the system does not maintain sufficient memory capacity). Thus, as soon as the repairperson arrives, a maintenance process is required to detect failed components, reinstall new

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access

devices and restart the overall system. The detection and maintenance processes are captured by the transition $T_{MEMdet2}$. After this transition fires, all the tokens in P_{MEMdn} are removed and deposited in P_{MEMrp2}. The tokens return to P_{MEMup} after the transition T_{MEMrp2} fires to indicate that the new installation of memory devices is completed. The overall system is restarted.

SRN model of APP subsystem

Fig. 9g presents the SRN model of the APP subsystem. We suppose that a host runs a number of applications at the beginning depicted by n_{APP} in the place P_{APPup}. Because at any time there are multiple applications in the running state competing to fail first, the failure of an application occurs with a failure rate depending on the number of the currently running applications. As an application fails, the timed transition T_{APPf} is fired and a token in P_{APPup} is removed and deposited in P_{APPdn}. The failed application is then halted and restarted to avoid further data corruption. The recovery of the failed application is depicted by firing the transition T_{APPr} and thus the aforementioned token in the place P_{APPdn} is removed and deposited in the running state P_{APPup}. An application enters an update process as the transition T_{APPudr} is fired and thus, a token in P_{APPup} is removed and deposited in $P_{APPupdate}$. The transition $T_{APPupdate}$ is triggered to fire in order to depict the update of an application. The update is completed as one token in $P_{APPupdate}$ is removed and deposited in P_{APPup}. The application returns to its operational state with new updates. In this model, we incorporate the dependence of the APP subsystem on the underlying systems, particularly the VM subsystem. As long as a VM that hosts an app is in uptime, one token remains in the place P_{APPdup}. When the VM fails, the transition T_{APPdf} is fired and the token in P_{APPdup} is removed and deposited in P_{APPddn}. As an aftermath, all apps are vanished immediately regardless of their current operational states. This behavior is captured by a set of immediate transitions including t_{APPupo}, t_{APPdno}, $t_{APPupdateo}$ to remove all tokens from the places P_{APPup}, P_{APPdn}, $P_{APPupdate}$, respectively. When the underlying VM subsystem returns to its uptime, the transition T_{APPdr} fires and the token in P_{APPddn} is removed and deposited in P_{APPdup}. The APP subsystem is now able to start the new apps running on the operational VMs. Thus, the transition T_{APPupi} is enable to fire and deposit tokens one at a time into the upstate P_{APPup}.

SRN models of VMM and VM subsystems

The operations of a VMM subsystem are captured as in *Fig. 9h*. In the beginning, the underlying hardware subsystems are supposed to operate in a healthy state as depicted by one token in the place P_{VMMdup}; and the VMM subsystem begins with n_{VMM} of VMMs in upstate P_{VMMup}. A VMM can fail due to regular faults as one token is removed from the place P_{VMMup}, then deposited in P_{VMMf} through the transition T_{VMMf}. When the transition T_{VMMfd} is fired, the token representing a failed VMM in the place P_{VMMf} is removed and transited to the place P_{VMMfd}, which indecates that the detection process of the failure of the VMM is triggered and proceeded. As soon as the faults causing the failure of the VMM are detected, a repairperson recovers the failed VMM. In the VMM model, this stage is captured by firing the transition T_{VMMfr}, then the token representing for the VMM in the fault detection process in the place P_{VMMfd} is removed and returned to the place P_{VMMup} (normal and healthy state).

In another scenario, a VMM likely experiences the problems associated with software aging and thus enters a failure-probable state in which the performance of the VMM degrades over time, and an aging failure can occurs with high probability. This phenomenon is captured by the transition of a token in the place P_{VMMup} (healthy state) to the place P_{VMMfp} (failure-probable state). The firing of the transition T_{VMMfp} with the rate of λ_{VMMfp} indeed represents the aging phenomenon of a VMM subsystem in modeling [61]. As the VMM is aging in the failure-probable state, if there is no proactive action, the VMM likely enters downtime due to an uncertain failure resulting from aging. The aging failure occurs when the transition T_{VMMaf} fires and one token representing the aged VMM in the place P_{VMMfp} is removed and deposited in the place P_{VMMaf}. As soon as a VMM fails because of aging, it is necessary to summon a repairperson to proceed with a detection process and then repair the aging faults. This stage often requires a comprehensive detection and repair and thus consumes time and efforts [81]. The detection and repair processes of aging failure are captured by the transition of the token represented for a failed VMM in the place P_{VMMaf} to the place P_{VMMafd} as the transition T_{VMMafd} fires; and then from the place P_{VMMafd} to the place P_{VMMup} through the fired transition T_{VMMafr}. The VMM, which has failed due to aging returns to its normal state. Aing problems can likely be overcome by using rejuvenation techniques [81]. Among the different techniques [76], we employ time-based rejuvenation to pro-actively and periodically purge aging bugs from the VMM subsystem. Accordingly, the rejuvenation of the VMM subsystem is implemented based on a time clock. The VMM clock starts counting time as the system begins to operate, which is indicated by a token in the place P_{VMMclk}. To approximate the transition of time, we use 10-stage Erlang distribution attached to the transition $T_{VMMclkinterval}$ (depicted as a large black rectangle) [61, 76]. After a time interval passes, the transition $T_{VMMclkinterval}$ fires, the token in the place P_{VMMclk} is removed and deposited in the place $P_{VMMclktrig}$.

This is to trigger the time-based rejuvenation procedures on the VMM subsystem regardless of its current states (normal or aging states). As a result, all the tokens either in the place P_{VMMup} (healthy state) or in the place P_{VMMfp} (failure-probable state) are instantly removed and deposited in the same place P_{VMMrej} through the fired immediate transitions $t_{VMMuprej}$ and $t_{VMMfprej}$. When the place P_{VMMrej} contains a token (representing a VMM in rejuvenation), the immediate transition t_{VMM} is enabled to transit the token in the place $P_{VMMclktrig}$ to the place $P_{VMMrejtrig}$, which is also the condition to enable the time transition T_{VMMrej} to allow the rejuvenation process of the VMMs to start. The tokens in the place P_{VMMrej} are sequentially removed and deposited in the place P_{VMMup} for those VMMs in the rejuvenation period return to a healthy state. When no VMM is being subjected to rejuvenation processes, the immediate transition $t_{VMMclkreset}$ fires to transit the token in the place $P_{VMMrejtrig}$ to the place P_{VMMclk} (a new interval for time-based rejuvenation restarts).

In this SRN model of the VMM subsystem, we also incorporate the dependency between the underlying hardware subsystems (CPU, MEM, NET, PWR, and COO as depicted in *Figs. 9b to 9f*). When the hardware subsystem (HW) is up, the place P_{VMMdup} contains one token. The system fails when one of its subsystems fails, in which case the token in the place P_{VMMdup} is removed and deposited in the place P_{VMMddn} through the enabled transition T_{VMMdf}. The dependency between the VMM subsystem and the hardware subsystem is captured in the way that when the hardware subsystem enters downtime (one token in the place P_{VMMddn} as mentioned above), all tokens in every places of the VMM and VMM clock models are removed instantly. This removal occurs through the immediate transitions correspondingly attached to the places via output arcs regardless of the current states of the VMM subsystem as follows t_{VMMupo}, t_{VMMfo}, t_{VMMfdo}, t_{VMMfpo}, t_{VMMafo}, $t_{VMMafdo}$, $t_{VMMrejo}$ (for VMM model) and $t_{VMMclko}$, $t_{VMMclktrigo}$, $t_{VMMrejtrigo}$ (for the VMM clock model). As soon as the hardware subsystem returns to the normal state (depicted by the transition of the token in the place P_{VMMddn} to the place P_{VMMdup} through the fired transition T_{VMMdr}), the VMM subsystem initiates its new VMM and the VMM clock starts counting a new interval for rejuvenation. This behavior is captured by enabling the time-transitions T_{VMMupi} (to initiate VMMs) and $T_{VMMclki}$ (to initiate clock token), respectively. The rates of the transitions T_{VMMdf} and T_{VMMdr} are computed upon the mean time to failure equivalent (MTTFeq) and mean time to recovery equivalent (MTTReq), respectively, of the HW subtree in the FT as depicted in Fig. 9a.

Fig. 9i shows a SRN model of the VM subsystem . In this model, we capture the failures due to non-Mandelbugs and aging problems as similarly as in the VMM model. The operations and behaviors of the subsystem are described in the VM model (Fig. 9i) in accordance with the previous description of the VMM model (Fig. 9h). The dependency captured in the VM model is actually the dependency between the VM subsystem and the hosting VMM subsystem (whereas the dependency in the VMM model is the dependency between the VMM subsystem and its underlying hardware subsystem). The rates (of the transitions T_{VMdf} and T_{VMdr}) in which the whole system under the VM subsystem enters downtime or returns to the normal state are computed correspondingly based on the MTTFeq and MTTReq of the VMM subsystem in Fig. 9h.

2) Modeling of a switch

i. *Architecture of a switch*

A switch is modeled using a two-level hierarchical model as depicted in Fig. 11. The availability modeling of a specific switch using RBD and CTMC is detailed in [27, 47, 73]. We use FT and SRN for consistency across the modeling of the entire system in order to model the switch with the same configuration. The architecture of our switch follows a distributed routing manner based on the architecture of the Cisco GSR 12000 (Cisco, San Jose, California) [82]. The architecture and functionalities of the switch (depicted as in the Fig. 10) in accordance with [27, 47, 83] are described briefly as follows:

- *Gigabit Route Processors (GRP)*[84]: A GRP as the brain of the switch runs protocols and computes the forwarding tables then distributes them to all line cards over the switch fabric. Furthermore, GRPs manage system control and the administrative functions of the switch (diagnosis, console port, and line card monitoring).
- *Line Cards (LC)*[85]: A LC (either the ingress or egress LC) performs packet forwarding, ping response, packet fragmentation (particularly including queuing, congestion control, statistics, and other features such as access lists and the committed access rate). GRPs distribute copies of most updated forwarding tables to each LC. An independent lookup of a destination address is then performed on each LC for each datagram received on a local routing table. The detailed architecture of a LC is described in [86].
- *Switch Fabric*[87]: (or multi-gigabit crossbar switch fabric) as the heart of the switch connects all LCs to each other through centralized point-to-point serial lines to provide high capacity switching at gigabit rates thereby enabling high performance of the switch.
  - *Switch Fabric Cards (SFC)*: enables multiple bus transactions in a simultaneous manner to provide multi-gigabit switching functions (as an NxN

matrix, where N is the number of LC slots)
  - *Clock and Scheduler Cards (CSC)*: synchronize LCs to transmit or receive data within any given fabric cycle and provide scheduling information and clocking reference to the SFC.
- *Internetworking Operating System (IOS)*: is a software package that integrates a variety of main functionalities within the switches (packet routing, switching, internetworking and telecommunications) and runs as a multitasking operating system on the switch.
- *Periodic Router Software Upgrade (Upgrade)*: A switch likely undergoes an outage when it needs a periodic software upgrade. Thus, we consider an upgrade as an event that affects the overall availability of the switch. In the modeling, we intentionally incorporate the upgrade event in a similar manner as in the other modules.
- *Chassis*[88]: All the components of the switch are installed on a chassis with a pre-designed configuration based on different versions of the switches. To simplify the modeling of the switch, we assume the chassis to be a non-redundant module which in turn consists of a maintenance bus, redundant power supplies, and a cooling system as a whole.
- *System Configuration*[27]: An 1:1 (1 primary and 1 standby) redundant scheme is employed for GRP and IOS, whereas a 1:N redundancy is applied for SFC in which one standby SFC is needed for every N SFCs. Further, at least one CSC with an additional one for reliability and performance are required.
- *Failure Modes*: LCs and GRP can fail due to a certain fault in either the hardware or software. SCS/SFC modules fail if they encounter a hardware fault. Meanwhile, IOS fails when a software fault occurs. The switch also stops running if it enters an upgrade process. The switch is available with at least four functional CSC/SFC modules.

ii. ***Modeling of a switch***:

Fault tree of a switch

(Fig. 11a): The overall failure of a switch is captured by a FT as in Fig. 11a, in which the individual failure of any node/module (including Upgrade, LC-in, LC-out, CSC-SFC, GRP, Chassis, and IOS) in consideration certainly causes the overall failure of the switch.

SRN model of periodic upgrade event

(Fig. 11b): Fig. 11b depicts the modeling of the upgrade process for the switch. When the switch is running in normal state, a token resides in the place P_{Unor}. After a certain period of time, the switch needs to upgrade its firmware. This enables the transition T_{Run} and deposits the token in the place P_{Unor} into the place P_{Uup}. When the upgrade process completes, the token in the place P_{Uup} is removed and deposited in the place P_{Unor} through the fired transition $T_{Upgrade}$. The switch returns to its normal state with updated firmware.

SRN model of chassis module

(Fig. 11c): A two-state (up and down) SRN model is used to simplify the modeling of the non-redundant chassis. When the chassis enters an outage from normal state (a token in the place P_{Cup}), the transition T_{Cf} is enabled to remove and deposit the token in the place P_{Cup} into the place P_{Cdn}. As soon as the recovery of the chassis is completed and the chassis returns to normal state, the token in the place P_{Cdn} is removed and deposited into the place P_{Cup} through the fired transition T_{Cr}.

SRN model of LC-in and LC-out

(Figs. 11d and 11e): LC-in and LC-out are the non-redundant modules which probably encounter failures either due to hardware or software. We also consider only two states (up and down) of each hardware or software. Thus, both LC-in and LC-out can be modeled similarly as a three-state SRN model. The model of LC-in (Fig. 11d) is explained, whereas the model of LC-out is referred to in the same way. Initially, a LC-in is operational with a token in the place P_{LCiup}. If the hardware fails, the transition T_{LCihf} is enabled to remove the token in the place P_{LCiup} and deposit it into the place P_{LCihd} (downstate of LC-in's hardware). Otherwise, the LC-in may fail due to software, in which case the transition T_{LCisf} is fired, and the token in the place P_{LCiup} is removed and deposited in the place P_{LCisd} (downstate of LC-in software). The recoveries of the LC-in hardware and software are captured by the firing of the transitions T_{LCihr} and T_{LCisr}, respectively. When these transitions fire, the token in either the places P_{LCihd} or P_{LCisd} is removed and deposited in P_{LCiup}. The LC returns to its healthy state after the recovery of the hardware or software.

SRN model of CSC-SFC modules

(Fig. 11f): The modules CSC and SFC are modeled together in a single model as in Fig. 11f to satisfy the constrain of the total number of operational devices (at least four out of five CSC/SFC modules are operational for the switch to be available). The model is initiated with two and three tokens respectively in the places P_{CSCup} and P_{SFCup} (normal states of the modules). The failure of a CSC occurs when the transition T_{CSCf} fires whereas if the transition T_{SFCf} is enabled, a SFC undergoes an outage. After the firing of these transitions, a token in the places P_{CSCup} or P_{SFCup} is removed and deposited in the places P_{CSCdn} or P_{SFCdn}, correspondingly. When multiple CSC/SFC cards are in the normal state, these cards tend to compete

with each other to fail first. The failure rates are therefore proportionally dependent on the number of running cards (which is the number of tokens in the corresponding places P_{CSCup} or P_{SFCup}). This marking dependence is implied by the ♯ sign next to the respective transitions T_{CSCf} and T_{SFCf}. The constrain for this composited model to be in the upstate is that the pair of numbers (*m*-*n*) (which represents the state in which *m* CSCs and *n* SFCs are up) must satisfy the condition: $m+n \geq 4$. This constrain is captured in the reward function to compute the metrics of interest for the CSC-SFC module in the overall hierarchical model.

SRN model of GRP/IOS modules

(Figs. 11g and 11h): The operations of the modules GRP and IOS are captured in Figs. 11g and 11h, respectively. Since both the modules GRP/IOS are a (*1:N*) redundant module (with *N* active units) in which their operational states are identical. We then describe the model of GRP in Fig. 11g, and the model for IOS in Fig. 11h is referred to accordingly. The model initiates a token in the state P_{GRPnor} to represent the normal state of all hardware components. Either of the active and standby units in the GRP can fail. Imperfect coverage is incorporated in the model to capture the failure detection processes without success. When an active unit fails and its failure detection also fails, the operational state of the GRP moves from P_{GRPnor} to P_{GRPafu}. Accordingly, the token in P_{GRPnor} is removed and deposited in P_{GRPafu} through the fired transition T_{GRPafu}. Nevertheless, if the failure of the active unit is detected successfully, the token in P_{GRPnor} is removed and instead deposited in P_{GRPafd}. The state transition rate of the transition T_{GRPafu} is $N.\lambda_3.(1-c_3)$ and it is $N.\lambda_3.c_3$ for the transition T_{GRPafd}, where λ_3 and c_3 are the failure rate of an individual unit and the coverage factor of an active unit, respectively. The repair of a failed active unit under unsuccessful detection occurs at the rate μ_4 when the transition $T_{GRPafur}$ is fired and subsequently the token in P_{GRPafu} is removed and deposited in P_{GRPnor}. The GRP module returns to its normal state. In the case of successful detection, the standby unit takes over the operations of the failed active unit at the rate of β_2. This switchover process is captured by the firing of the transition T_{GRPstd}. The token in P_{GRPafd} is then removed and deposited in P_{GRPstd}. At this point, if the next active unit fails with the rate of $N.\lambda_3$ while trying to recover the first active unit, the state of the module changes to P_{GRPa2f}. The transition T_{GRPa2f} is fired to remove the token in P_{GRPstd} and deposit it in P_{GRPa2f}.

VI. NUMERICAL AND SIMULATION RESULTS

The models are all implemented in SHARPE [58]. The default values of the parameters using in the following analyses are summarized in Table 1, based on previous works [76, 89, 90].

Complex interconnection in networks often comes along with the existence of multiple redundant paths between pairs of network elements. For this reason, ones can still find a continuous connection between a certain pair even in the case of the failure of some network elements [47]. *The network reliability/availability is often defined as the probability a designated pair of nodes of the graph are connected through at least one path of working edges as first presented in [91–93].* In the graph abstraction of multi-valued networks, the network elements can have multiple states. In particular, we defined and obtained the reliability/availability of the lowest components in the network using reward functions as defined in Table 3 as follows:

- *Switch:* is available if all of its components are operational, in other words, a switch fails if at least one of its components fails as shown in Fig. 11a
  - *Upgrade:* is available if it is in a normal state captured by a token in P_{Unor}.

$$ssa_{Upgrade} = \begin{cases} 1 : \#(P_{Unor}) > 0 \\ 0 : otherwise \end{cases} \quad (11)$$

  - *Chassis:* is available if it is in a normal state captured by a token in P_{Cup}.

$$ssa_{Chassis} = \begin{cases} 1 : \#(P_{Cup}) > 0 \\ 0 : otherwise \end{cases} \quad (12)$$

  - *LCin:* is available if it is in operational state captured by a token in P_{LCiup}.

$$ssa_{LCin} = \begin{cases} 1 : \#(P_{LCiup}) > 0 \\ 0 : otherwise \end{cases} \quad (13)$$

  - *LCout:* is available if it is in operational state captured by a token in P_{LCoup}

$$ssa_{LCout} = \begin{cases} 1 : \#(P_{LCoup}) > 0 \\ 0 : otherwise \end{cases} \quad (14)$$

  - *CSC_SFC:* is available if the total number of normal components of CSC/SFC are not less than four, represented by the total number of tokens in the two up states P_{CSCup} and P_{SFCup}.

$$ssa_{CSC\_SFC} = \begin{cases} 1 : \#(P_{CSCup}) + \\ \#(P_{SFCup}) >= 4 \\ 0 : otherwise \end{cases} \quad (15)$$

  - *GRP:* is available if it is in any of its up states as defined in Table 4 including P_{GRPnor}, P_{GRPafd}, P_{GRPstd}, and P_{GRPsfu}.

$$ssa_{GRP} = \begin{cases} 1 : \#(P_{GRPnor}) + \#(P_{GRPafd}) + \\ \#(P_{GRPstd}) + \#(P_{GRPsfd}) + \\ \#(P_{GRPsfu}) > 0 \\ 0 : otherwise \end{cases} \quad (16)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access

– *IOS:* is available if it is in any of its up states as defined in Table 4 including P_{IOSnor}, P_{IOSafd}, P_{IOSstd}, P_{IOSsfd} and P_{IOSsfu}.

$$ssa_{IOS} = \begin{cases} 1 : \#(P_{IOSnor}) + \#(P_{IOSafd}) + \\ \#(P_{IOSstd}) + \\ \#(P_{IOSsfd}) + \\ \#(P_{IOSsfu}) > 0 \\ 0 : otherwise \end{cases} \quad (17)$$

- *Host:* is available if all its hardware and software components are available.
  – *CPU:* is available if the number of CPUs in normal state is not smaller than a pre-defined positive number, which is captured by the number of tokens in P_{CPUnor}.

$$ssa_{CPU} = \begin{cases} 1 : \#(P_{CPUnor}) >= m_{CPU} \\ 0 : otherwise \end{cases} \quad (18)$$

  – *MEM:* is available if the number of operational MEMs is not smaller than a pre-defined positive number, which is captured by the number of token in P_{MEMup}.

$$ssa_{MEM} = \begin{cases} 1 : \#(P_{MEMup}) >= m_{MEM} \\ 0 : otherwise \end{cases} \quad (19)$$

  – *NET:* is available if the number of operational NETs is not smaller than a pre-defined positive number, which is captured by the number of token in P_{NETup}.

$$ssa_{NET} = \begin{cases} 1 : \#(P_{NETup}) >= m_{NET} \\ 0 : otherwise \end{cases} \quad (20)$$

  – *PWR:* is available if the number of operational PWRs is not smaller than a pre-defined positive number, which is captured by the number of token in P_{PWRup}.

$$ssa_{PWR} = \begin{cases} 1 : \#(P_{PWRup}) >= m_{PWR} \\ 0 : otherwise \end{cases} \quad (21)$$

  – *COO:* is available if the number of operational COOs is not smaller than a pre-defined positive number, which is captured by the number of token in P_{COOup}.

$$ssa_{COO} = \begin{cases} 1 : \#(P_{COOup}) >= m_{COO} \\ 0 : otherwise \end{cases} \quad (22)$$

  – *VMM:* is available if it is either in normal state or failure-probable state captured by a token in either P_{VMMup} or P_{VMMfp}.

$$ssa_{VMM} = \begin{cases} 1 : \#(P_{VMMup}) + \\ \#(P_{VMMfp}) > 0 \\ 0 : otherwise \end{cases} \quad (23)$$

  – *VM:* is available if it is either in normal state or failure-probable state captured by a token in either P_{VMup} or P_{VMfp}.

$$ssa_{VM} = \begin{cases} 1 : \#(P_{VMup}) + \\ \#(P_{VMfp}) > 0 \\ 0 : otherwise \end{cases} \quad (24)$$

  – *APP:* is available if the number of running APPs is not smaller than a pre-defined positive number of APPs for running services to be available captured by the number of tokens in P_{APPup}.

$$ssa_{APP} = \begin{cases} 1 : \#(P_{APPup}) >= m_{APP} \\ 0 : otherwise \end{cases} \quad (25)$$

- *Links:* represent the physical inter-connections between network elements (switches and servers). We assume to not take into account sophisticated behaviors of links in a network and we charactertize the links by using normal state (connected and thus, available) and abnormal state (disconnected and thus, unavailable) which are captured by the two parameters MTTF (λ_L) and MTTR (μ_L) of links.

A. RELIABILITY ANALYSIS

Fig. 12a and Fig. 12b illustrate the dynamic behaviors of the overall system reliabilities (as a function of time) for three-tier and fat-tree DCNs in all cases when the respective network configurations alter. In general, the reliability of the system in all cases decays according to a certain varying curve as time proceeds with different speeds depending on the corresponding network configurations. This also shows the effects of compute node routing policies on the overall reliability.

In the case of three-tier DCN, the reliability decay of all case-studies follows in similar variation curves with not clear difference when we look at the whole decaying period of time. But, when we look into a small time slice (for instance, [79.5-80.5] hours), we can see the difference of reliability in accordance with routing topologies of the network. The most distributed routing topology (case VI) outperforms the highest level of reliability while the least distributed ones (case I and case II) show the least level of reliability. The cases IV and V obtain a similar level of reliability due to their similar routing policies.

In the case of fat-tree DCN (Fig. 12b) at a certain time slice, we find that the original networking system (case I - RG01) has the lowest reliability in comparison with all the remaining cases. Furthermore, as we magnify the plot in the time slice [79.5-80.5] hours, we observe clear differences of the reliabilities in the remaining cases (II-VI). The case VI (RG06) performs with the highest reliability in the selected time slice and at all time. As we compare the cases in the pairs (VI-V), (IV-II) and (III-I) (in which, the former cases clearly perform with higher reliability compared to the later cases in the same pair), we observe that the scattering and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access

allocation policies of the nodes within a pod enhance the overall reliability of the system. In contrast, if we observe the pairs (VI-IV), (IV-III), (V-II) and (II-I) we also conclude that the above-mentioned node scattering policies in different pods also improve the system reliability. We find that the scattering of nodes within a pod is only slightly effective compared to nodes in different pods to improve the system reliability.

As we compare the two figures, we find that the reliability of fat-tree DCN is lower than that of three-tier DCN at a certain time slice. Also, the reliability decaying curves of the former have higher slopes than those of the latter one do, that is to say the reliability of fat-tree DCN decays faster than that of three-tier DCN. Nevertheless, the both DCNs have a common concensus, that is the dispersion of active compute nodes can actually enhance the overall system reliability. System design usually requires strict selection of more reliable and expensive components rather than less reliable and inexpensive ones to achieve the predefined dependability. Nevertheless, in some cases, to require an increase in system reliablity, it is not an essential option to acquire highly robust devices. Generally, there are always global policies to enable the system designer to improve the overall system reliablity. The above results can facilitate guiding that purpose of system design.

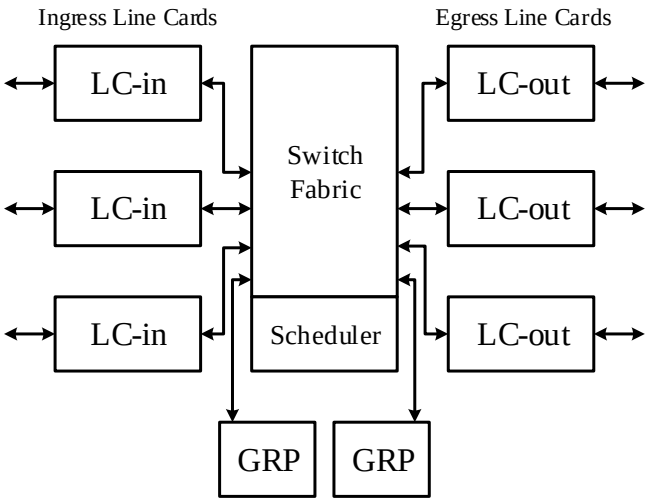
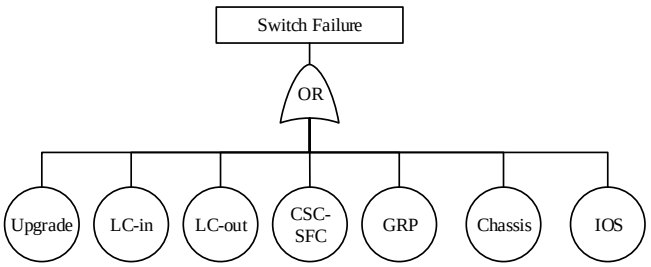
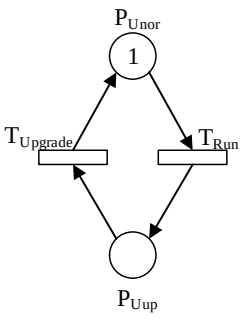


Figure 10: Architecture of a Switch

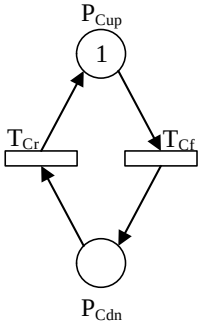
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access



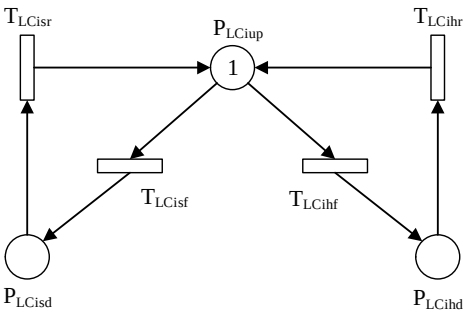
(a) Fault Tree of a Switch



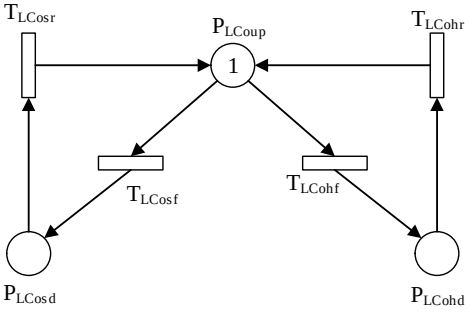
(b) Upgrade



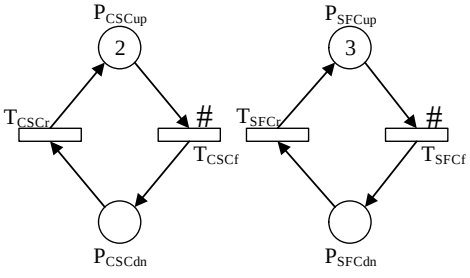
(c) Chassis



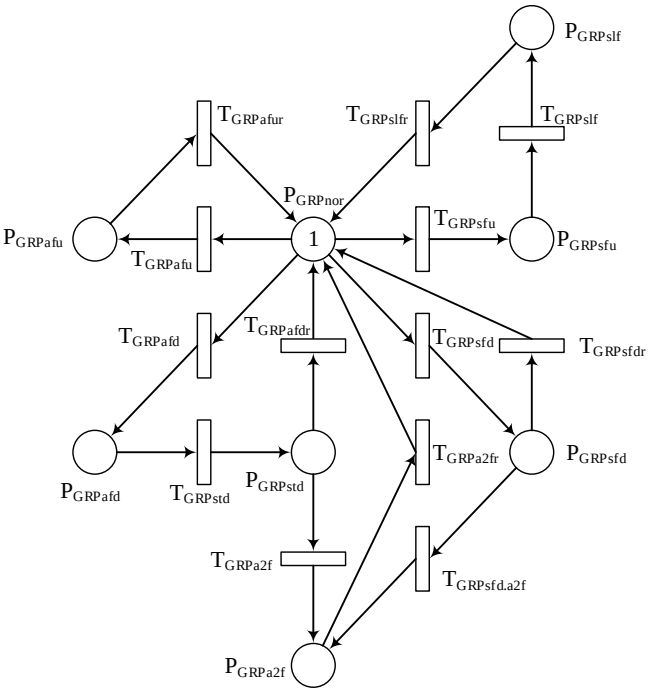
(d) LC-in



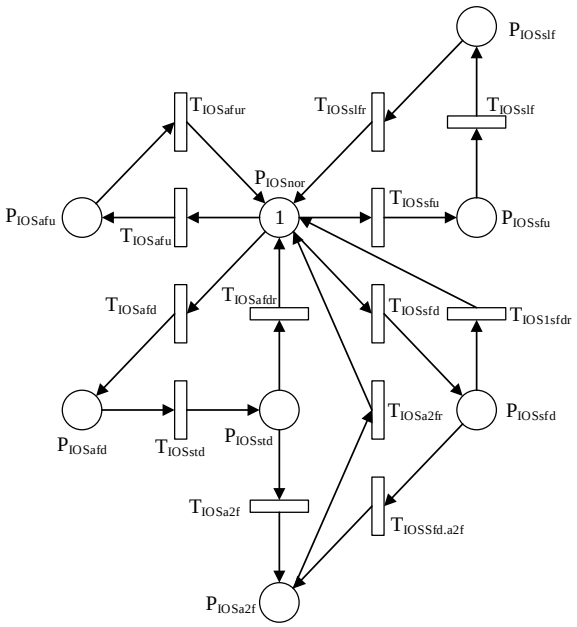
(e) LC-out



(f) CSC-SFC



(g) GRP



(h) IOS

Figure 11: Sub-models of a Switch

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access

Table 1: DEFAULT INPUT PARAMETERS OF SRN AND RG MODELS

Name	Attached Transition	Description	Meantime/Value	
—HOST: CPU—				
n_{CPU}		Number of CPUs on a Host	4	
m_{CPU}		Minimal number of CPUs to be available	2	
$1/\lambda_{CPU}$	T_{CPUf}	Mean time to failure of a CPU	1	year
$1/\mu_{CPU}$	T_{CPUrp}	Mean time to repair of a CPU	2	hours
$1/\beta_{CPUdet}$	T_{CPUdet}	Mean time to detection of a CPU failure	8	hours
—HOST: MEM—				
n_{MEM}		Number of MEMs on a Host	4	
m_{MEM}		Minimal number of MEMs to be available	2	
$1/\lambda_{MEM}$	T_{MEMf}	Mean time to failure of a MEM	90	days
$1/\mu_{MEM1}$	T_{MEMrp1}	Mean time to recovery of a MEM in the first scenario	2	hours
$1/\mu_{MEM2}$	T_{MEMrp2}	Mean time to recovery of a MEM in the second scenario	3	hours
$1/\beta_{MEMdet1}$	$T_{MEMdet1}$	Mean time to failure detection of a MEM in the first scenario	30	mins
$1/\beta_{MEMdet2}$	$T_{MEMdet2}$	Mean time to failure detection of a MEM in the second scenario	45	mins
—HOST: NET—				
n_{NET}		Number of NETs on a Host	4	
m_{NET}		Minimal number of NETs to be available	2	
$1/\lambda_{NET}$	T_{NETf}	Mean time to failure of a NET	30	days
$1/\mu_{NET1}$	T_{NETrp1}	Mean time to recovery of a NET in the first scenario	20	mins
$1/\mu_{NET2}$	T_{NETrp2}	Mean time to recovery of a NET in the second scenario	45	mins
$1/\beta_{NETdet1}$	$T_{NETdet1}$	Mean time to failure detection of a NET in the first scenario	2	hours
$1/\beta_{NETdet2}$	$T_{NETdet2}$	Mean time to failure detection of a NET in the second scenario	2.5	hours
—HOST: PWR—				
n_{PWR}		Number of PWRs on a Host	4	
m_{PWR}		Minimal number of PWRs to be available	2	
$1/\lambda_{PWR}$	T_{PWRf}	Mean time to failure of a PWR	30	days
$1/\mu_{PWR1}$	T_{PWRrp1}	Mean time to recovery of a PWR in the first scenario	30	mins
$1/\mu_{PWR2}$	T_{PWRrp2}	Mean time to recovery of a PWR in the second scenario	45	mins
$1/\beta_{PWRdet1}$	$T_{PWRdet1}$	Mean time to failure detection of a PWR in the first scenario	15	mins
$1/\beta_{PWRdet2}$	$T_{PWRdet2}$	Mean time to failure detection of a PWR in the second scenario	30	mins
—HOST: COO—				
n_{COO}		Number of coolers on a Host	4	
m_{COO}		Minimal number of coolers to be available	2	
$1/\lambda_{COO}$	T_{COOf}	Mean time to failure of a cooler	3	weeks
$1/\mu_{COO1}$	T_{COOrp1}	Mean time to repair coolers in the first scenario	45	minutes
$1/\mu_{COO2}$	T_{COOrp2}	Mean time to repair coolers in the second scenario	50	minutes
$1/\beta_{COOdet1}$	$T_{COOdet1}$	Mean time to detect failures of cooler of first scenario	15	minutes
$1/\beta_{COOdet2}$	$T_{COOdet2}$	Mean time to detect failures of cooler of second scenario	30	minutes
—HOST: VMM—				
n_{VMM}		Number of VMM on a Host	1	
n_{VMMclk}		Number of Erlang steps for VMM clock	10	
$1/\lambda_{VMMf}$	T_{VMMf}	Mean time to failure of a VMM	2654	hours
$1/\lambda_{VMMfp}$	T_{VMMfp}	Mean time to aging of a VMM	30	days
$1/\lambda_{VMMaf}$	T_{VMMaf}	Mean time to aging failure of a VMM	7	days
$1/\omega_{VMMrej}$	T_{VMMrej}	Mean time to rejuvenation of a VMM	15	mins
$1/\omega_{VMMclk}$	$T_{VMMclkinterval}$	Mean time to clock trigger of a VMM	7	days
$1/\mu_{VMMr}$	T_{VMMr}	Mean time to recovery of a VMM's normal failure	100	mins
$1/\mu_{VMMafr}$	T_{VMMfr}	Mean time to recovery of a VMM's aging failure	65	mins
$1/\beta_{VMMafd}$	T_{VMMafd}	Mean time to detect a VMM failure due to aging	2	hours
$1/\beta_{VMMfd}$	T_{VMMfd}	Mean time to detect a common failure of a VMM	30	mins
$1/\omega_{VMMupin}$	$T_{VMMupin}$	Mean time to create and start a new VMM	15	mins
—HOST: VM—				
n_{VM}		Number of VM on a Host	4	
n_{VMclk}		Number of Erlang steps for VM clock	10	
$1/\lambda_{VMf}$	T_{VMf}	Mean time to failure of a VM	2	days
$1/\lambda_{VMfp}$	T_{VMfp}	Mean time to aging of a VM	8	hours
$1/\lambda_{VMaf}$	T_{VMaf}	Mean time to aging failure of a VM	3	hours
$1/\omega_{VMrej}$	T_{VMref}	Mean time to rejuvenation of a VM	10	mins
$1/\omega_{VMclk}$	$T_{VMclkinterval}$	Mean time to clock trigger of a VM	1	day
$1/\mu_{VMr}$	T_{VMr}	Mean time to recovery of a VM's normal failure	30	mins
$1/\mu_{VMafr}$	T_{VMfr}	Mean time to recovery of a VM's aging failure	120	mins

Continued on next page

Continued from previous page

Name	Attached Transition	Description	Meantime/Value	
$1/\beta_{VMafd}$	T_{VMafd}	Mean time to a VM failure due to aging	45	mins
$1/\beta_{VMfd}$	T_{VMfd}	Mean time to a common failure of a VM	20	mins
$1/\omega_{VMupin}$	T_{VMupin}	Mean time to create and start a new VM	5	mins
—HOST: APP—				
n_{APP}		Number of Apps on a Host	4	
m_{APP}		Minimal number of Apps to be available	2	
$1/\lambda_{APPf}$	T_{APPf}	Mean time to failure of an APP	8	hours
$1/\mu_{APPr}$	T_{APPr}	Mean time to recovery of an APP update	30	minutes
$1/\omega_{APPudr}$	T_{APPudr}	Mean time to an APP update	3	days
$1/\omega_{APPupdate}$	$T_{APPupdate}$	Mean time to completion of an APP update	20	minutes
$1/\omega_{APPupin}$	$T_{APPupdate}$	Mean time to create and start an APP	30	seconds
—SWITCH: Upgrade—				
$MTBU$	T_{Run}	Mean-time-between-upgrade	4320	hours
$MTTU$	$T_{Upgrade}$	Mean-time-to-upgrade	20	mins
—SWITCH: LC-in—				
$1/\lambda_{Sin}$	T_{LCisf}	Mean time to software failure of LC-in	18000	hours
$1/\mu_{Sin}$	T_{LCisr}	Mean time to software repair of LC-in	20	mins
$1/\lambda_{Hin}$	T_{LCihf}	Mean time to hardware failure of LC-in	111050	hours
$1/\mu_{Hin}$	T_{LCihr}	Mean time to hardware repair of LC-in	2	hours
—SWITCH: LC-out—				
$1/\lambda_{Sout}$	T_{LCosf}	Mean time to software failure of LC-out	18000	hours
$1/\mu_{Sout}$	T_{LCosr}	Mean time to software repair of LC-out	20	mins
$1/\lambda_{Hout}$	T_{LCohf}	Mean time to hardware failure of LC-out	103402	hours
$1/\mu_{Hout}$	T_{LCohr}	Mean time to hardware repair of LC-out	2	hours
—SWITCH: CSC-SFC—				
$1/\lambda_1$	T_{CSCf}	Mean time to failure of CSC	272584	hours
$1/\mu_1$	T_{CSCr}	Mean time to repair of CSC	2	hours
$1/\lambda_2$	T_{SFCf}	Mean time to failure of SFC	422115	hours
$1/\mu_2$	T_{SFCr}	Mean time to repair of SFC	2	hours
—SWITCH: GRP—				
$1/\lambda_{GRP}$	T_{GRPsfu}, T_{GRPslf}, T_{GRPafu}, $T_{GRPafdr}$, T_{GRPsfd}, T_{GRPa2f}, $T_{GRPsfd.a2f}$	Mean time to failure of GRP	108304	hours
$1/\mu_{GRP}$	$T_{GRPslfu}$, $T_{GRPafur}$, $T_{GRPafdr}$, $T_{GRPa2fr}$, $T_{GRPsfdr}$	Mean time to repair of GRP	2	hours
$1/beta_{GRP}$	T_{GRPsfd}	Meantime to switchover of GRP	20	mins
c_3		Coverage of GRP active failure	0.9	
c_4		Coverage of GRP standby failure	0.6	
—SWITCH: Chassis—				
$1/\lambda_C$	T_{Cf}	Mean-time-to-failure of Chassis	396510	hours
$1/\mu_C$	T_{Cr}	Mean-time-to-repair of Chassis	4	hours
—SWITCH: IOS—				
$1/\lambda_{IOS}$	T_{IOSsfu}, T_{IOSslf}, T_{IOSafu}, $T_{IOSafdr}$, T_{IOSsfd}, T_{IOSa2f}, $T_{IOSsfd.a2f}$	Mean-time-to-failure of IOS	18000	hours
$1/\mu_{IOS}$	$T_{IOSslfu}$, $T_{IOSafur}$, $T_{IOSafdr}$, $T_{IOSa2fr}$, $T_{IOSsfdr}$	Mean-time-to-repair of IOS	20	mins
$1/beta_{IOS}$	T_{IOSsfd}	Mean time to switchover of IOS	20	mins
c_5		Coverage of IOS active failure	0.9999	
c_6		Coverage of IOS standby failure	0.9999	
—Link—				

Continued on next page

Continued from previous page

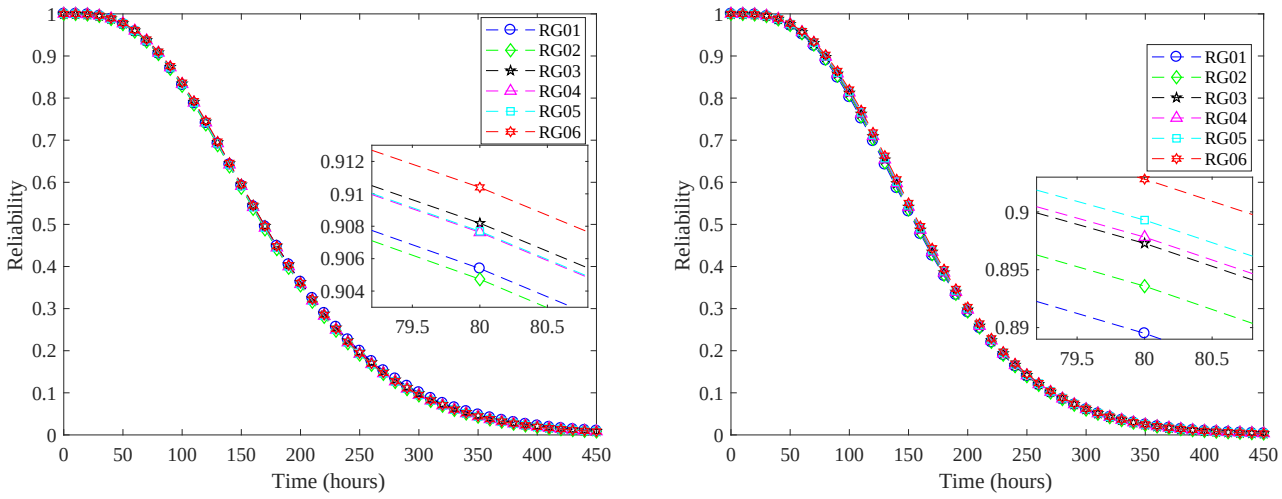
Name	Attached Transition	Description	Meantime/Value
$1/\lambda_L$		Mean time to failure of a connection between components	30 days
$1/\mu_L$		Mean time to repair of a connection between components	8 hours

Concluded

Table 2: STEADY STATE AVAILABILITY ANALYSIS OF DCNS UNDER DEFAULT PARAMETERS

Systems	MTTFeq	MTTReq	SSA	Number of nines
CPU	6.69940189E+08	1.61904761E+01	9.99999976E-01	7.61979
MEM	1.65310605E+08	3.71198414E+00	9.99999978E-01	7.65758
NET	4.98578505E+06	5.35141450E+00	9.99998927E-01	5.96940
PWR	6.53234116E+07	1.21352383E+00	9.99999981E-01	7.72125
COO	1.32538139E+07	1.45904442E+00	9.99999890E-01	6.95861
VMM	5.83920199E+05	9.96304600E+01	9.99829406E-01	3.76804
VM	4.25878967E+03	3.04937544E+01	9.92890711E-01	2.14817
APP	1.87718579E+02	2.30813219E+00	9.87853643E-01	1.91555
HOST	1.79728652E+02	3.54409951E+00	9.80662158E-01	1.71359
Upgrade	4.32000000E+03	3.33333333E-01	9.99922845E-01	4.11264
LC-in	1.54893452E+04	5.65801369E-01	9.99963473E-01	4.43739
LC-out	1.53311807E+04	5.80446231E-01	9.99962141E-01	4.42183
CSC-SFC	5.73352686E+09	2.58546360E+00	9.99999999E-01	9.00000
GRP	3.03241001E+05	2.00000000E+00	9.99993405E-01	5.18079
Chassis	3.96510000E+05	4.00000000E+00	9.99989912E-01	4.99619
IOS	7.59507185E+07	3.33333340E-01	9.99999996E-01	8.39794
ROUTER	2.72404159E+03	4.58312949E-01	9.99831781E-01	3.77412

	Three-tier			Fat-tree		
	SSA	#(nines)	Downtime (mins/year)	SSA	#(nines)	Downtime (mins/year)
RG01	0.999997016180	5.525227	1.56829560	0.999996139672	5.41337	2.02898845
RG02	0.999997055015	5.530917	1.54788437	0.999996913052	5.51046	1.62249979
RG03	0.999997384604	5.582462	1.37465230	0.999997251489	5.56083	1.44461723
RG04	0.999997384995	5.582528	1.37444640	0.999997273963	5.56447	1.43280484
RG05	0.999997385466	5.582606	1.37419918	0.999997279243	5.56531	1.43002947
RG06	0.999997639740	5.627040	1.24055283	0.999997544296	5.60982	1.29071781



(a) Three-Tier (b) Fat-Tree

Figure 12: Reliability of DCNs

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access

B. AVAILABILITY ANALYSIS

1) Steady State Availability Analysis

Steady state analyses are very much important, specifically in availability assessment of a certain system. System availability in steady state is also called long-term availability or asymptotic availability. This is to say that, in practice, the system availability will start approaching steady state availability (SSA) after a (long) period of time depending on maintainability and complexity of the system. Roughly saying that, SSA is a stabilizing baseline where the system availability is approximately a constant value. SSA is an important metric in system evaluation, especially for physical infrastructures, to predict future quality level of service that the physical system can deliver to its end-user. In this paper, we focus on SSA analyses of the DCNs in consideration rather than other metrics of performance. Because, we pay more careful attention on the capability and quality level of services that the physical infrastructure in cloud DC can fundamentally deliver to end-users as we look at the initial configurations of the system without any consideration of operational loads yet.

We evaluated the SSA of the systems, by attempting to compute the MTTFeq and MTTReq of every subsystem. We also compute the SSAs of all the subsystems. The number of nines in each SSA, respectively, is calculated for the sake of intuitive understanding. We computed SSA with number of nines and downtime minutes in a year for all case-studies of the both DCNs. The results are shown in Table 2. We find that the MTTFeqs of hardware subsystems (e.g. the CPU, and MEM in a HOST) to be much higher than that of the software based subsystems (VMM, VM or APP in a HOST). The MTTReqs of those hardware subsystems, however, differ in minor respects from those of the software subsystems. Therefore, the SSAs of the former subsystems are clearly higher than those of the latter ones.

In both cases of three-tier and fat-tree topologies, the results apparently pinpoint the effects of the compute node routing and allocation policies applied in this work, in which the scattering policies of compute nodes either within or between a pod apparently improve the system's SSA as compared to the original case with no specific policy (RG01). As we compare the SSAs of the case-studies, the most scattered routing topologies (case VI of three-tier and case VI of fat-tree) achieve the highest SSA, thus the lowest downtime minutes in a year. And the least distributed routing topologies (case I and case II in both DCNs) perform the lowest level of SSAs. In a comparison between the two three-tier and fat-tree routing topologies, we find that, the former obtains relatively higher values in the SSA analyses than the later does. The reason of this could be that, the three-tier routing provides more connectivity opportunities for the continuous connection between the four fixed servers and the four non-fixed ones. Moreover, the network configurations with scattering policies help the systems to achieve five-nines according to the requirements of high-availability industry standards.

2) Sensitivity Analysis

Sensitivity analyses with regard to major impacting parameters are described in this subsection. The sensitivity analyses are based on the sensitive variations of MTTFeqs and MTTReqs of the hosts, switches and links in the network systems. Those MTTFeqs and MTTReqs can be computed through the models in subsystem level of hosts, switches and links if there is any change in the architecture of these network elements. In this study, the internal architectures of the network elements are assumed to be fixed and only the parameters changes. Therefore, we do not need to re-model the entire infrastructure but only need to vary the values of parameters to observe the corresponding variation of the system availability. By analyzing the availability sensitivity of the network, we could explore the bottlenecks and the significant factors to gain the highest-available values of the system's overall availability.

Fig. 13 and Fig. 14 present the results of the sensitivity analysis of the system availability with respect to MTTF and MTTR respectively of hosts, switches and links in a detailed comparison among the case-studies. The figures pinpoint the effective impact of the compute node routing policies of active nodes in the network versus the original one. Furthermore, it also shows that the MTTF and MTTR of hosts are major parameters to boost the system availability.

Fig. 13a, Fig. 13c and Fig. 13e are the sensitivity analysis results of SSA for three-tier DCN with respect to $1/\lambda_h$, $1/\lambda_{sw}$ and $1/\lambda_l$. Accordingly, the variation of SSAs of all case-studies conforms with a common curve shape, in which in the range of small values (*<500 hours*), the SSAs are highly sensitive with respect to MTTFs of hosts, switches and links in the way that a small increase in value of those variables can lead to a huge improvement of the SSAs. When the MTTFs increase in the ranges of much bigger values, the SSAs gradually approach steady values. More specifically, we find that the most distributed routing (case VI - RG06) outperforms to obtain the highest level of SSA when MTTFs of hosts, switches and links change. While, the least distributed ones (case I - RG01 and case II - RG02) perform the lowest SSAs at any value of MTTFs. In a comparison between the MTTFs in term of their impact on the system's overall SSA, (comparing the results among the figures Fig. 13a, Fig. 13c and Fig. 13e), the MTTF of hosts has a significant impact since the change of the hosts' MTTF under default values of other parameters contributes a high value of SSA to the system (six numbers of nines). On the other hand, the MTTF of links is apparently the most sensitive in boosting the SSA especially when the values of the links' MTTF are in the small-value ranges (*<500 hours*) (depicted by the vertical graphs in Fig. 13e). Furthermore, the routing policy to multiple compute nodes which are connected to the same switches does not help gain higher availability in data connection in comparison to other cases as shown by the lower position of the graph representing

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access

RG02 in all figures.

Fig. 13b, Fig. 13d and Fig. 13f show the variation of SSA respectively for $1/\lambda_h$, $1/\lambda_{sw}$ and $1/\lambda_l$ in the sensitivity analysis of fat-tree DCN. In general, these images show a common feature of the dependency of the SSAs on the MTTFs respectively of hosts, switches, or links. That is, (i) when MTTFs increase in small ranges *(<500 hours)* , the value of SSA increases rapidly; (ii) whereas, for larger values *(>500 hours)* of MTTFs, the value of SSA gradually approaches a certain stabilized value. Furthermore, the differences in variation of SSAs when comparing the graphs show the sensitiveness of the MTTFs onto SSAs in which the MTTF of the links is most sensitive to SSA, then the MTTF of the switches, and finally the host MTTF. More specifically, a slight change in the MTTF value of links causes a huge change in the value of the SSA of the system. Especially when comparing SSA values at different ranges of MTTFs for each case (RG01-RG06) in each figure, we find that, as soon as the compute nodes of the system start being distributed to other pods (RG02-RG06 cases), the SSA of the system is significantly increased in comparison to the original case (RG01) (expressed as the distance between graphs in each figure). The distribution of the compute nodes in the network clearly impacts when the host MTTF changes. As we take a closer look at the enlarged graphs, we discover that the different distributions of the compute nodes also have different impacts on the SSA. In addition, the variations in the MTTFs of hosts, switches and links also change the SSA in the considered cases in different ways. More specifically, when the compute nodes are scattered all over the pods, the SSA always reaches the highest value regardless of any value of the MTTFs (shown as graphs for RG06 situated in a higher position than the ones for all other cases do). Conversely, when multiple nodes are connected with an edge-switch or in the same pod, the system obtains a significantly lower SSA. This is reflected in the position of the graph of the case RG02, which is always below the graphs of other cases.

When comparing the impact of MTTFs among the two topologies (three-tier and fat-tree), ones may find a consistence in which the MTTF of hosts has a significant contribution in vastly improving the system availability, while the MTTF of links is very much sensitive in impacting the system availability if its values are small.

Fig. 14a, Fig. 14c and Fig. 14e show the analysis results of SSAs with respect to $1/\mu_h$, $1/\mu_{sw}$ and $1/\mu_l$ in the case of three-tier topology. In general, we find that, when it comes to longer time to repair a failure of the above elements in a DCN (higher values of MTTRs), the system availability significantly drops down in a rapid manner. More specifically, when comparing the case-studies (RG01-RG06), the most distributed routing of compute nodes apparently outperforms to achieve the highest values of SSA at any slices of MTTR values (shown by the highest graph of RG06 in all the subfigures). On the contrary, the least scatterring ones perform the lowest level of SSA as depicted by the lowest graphs of RG01 and RG02. When comparing the impacts of the MTTRs, we see that the MTTR of hosts is more sensitive in changing the values of SSA since a little increase of the hosts' MTTR leads to a huge drop of the network's SSA in comparison to the other factors.

Fig. 14b, Fig. 14d and Fig. 14f present the variation in SSA of the system by the variables $1/\mu_h$, $1/\mu_{sw}$ and $1/\mu_l$ of fat-tree DCNs, respectively. Basically, when it takes longer for the system components to be repaired and restored after a failure (i.e., the MTTRs increase) the SSAs of the system can also be expected to decline in all considered cases. In addition, the more the value of the MTTRs rises, the more rapid the value of the SSA decreases (as shown by the slope of the graphs). Furthermore, a comparison of the graphs of different cases in the same figure indicates the effect of the distribution of compute nodes on the SSA improvement of the system. Similar to the above-mentioned descriptions, when the compute nodes begin to be distributed to other free nodes or to other pods or edge-switches in the same pod, we recognize that the SSA of the system has improved significantly. More specifically, the original case of RG01 always results in the lowest SSA, whereas the case in which the nodes are most widely distributed (RG06) always reaches the highest SSA in comparison to all the remaining cases. When we consider a specific range of MTTRs (enlarged graphs) more carefully, we also conclude that the way to allocate nodes in order to increase the number of connections to different edge-switches in different pods or different aggregation-switches in different pods specifically enhance the value of SSA. This can be seen when we compare the graphs for the case RG01, RG02 with the graphs of the remaining cases, especially RG06. When comparing the figures, we find that the MTTRs of the components also have different effects along with the distribution of compute nodes on the SSA of the system. Specifically, the MTTRs of the hosts have a more sensitive effects on the SSA than the MTTRs of switches and links. The MTTRs of switches have the least effect on the SSA. This is shown by the slope of the graphs in the figures. In Fig. 14b, we can see that only a little increase in the quantity of the MTTR of the host also significantly reduces the SSA of the system, whereas this change is small in Fig. 14f (especially, in the cases in which the compute nodes are scattered more widely). The above analysis results show that the compute nodes need to be scattered such that (i) the system configuration contains the number of connections between the nodes and the edge-switches as much as possible; (ii) or in different cases in which the numbers of connections to the edge-switches are the same, the compute nodes are able to connect to a larger number of aggregation-switches as many as possible.

When comparing the impact of MTTRs on the SSAs of both three-tier and fat-tree DCNs, we realize that the impact of hosts' MTTRs in the both topologies is similar to each other while the MTTRs of switches and links in the fat-tree DCN are more critical factors in impacting the system SSA in comparison to those in the three-tier DCN, shown by the

higher slope of graphs in Fig. 14d and Fig. 14f compared to that of graphs in Fig. 14c and Fig. 14e, respectively. This emphasizes the importance of recovery and maintenance of switches and links in fat-tree DCNs rather than those in three-tier DCNs.

In conclusion, the MTTFs of links and MTTRs of hosts are important parameters of the DCNs. This means that to enhance the SSA of the system we need to maintain the connection between components as long as possible to avoid errors and failures and at the same time we need to improve the time required to repair failed hosts as much as possible. Especially, in the operations and managements of the system, the system administrator should allocate and scatter compute nodes appropriately to achieve the highest SSA. Furthermore, depending on the chosen routing topologies, ones need to consider proper recovery and maintenence of more critical system elements to enhance system availability.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access

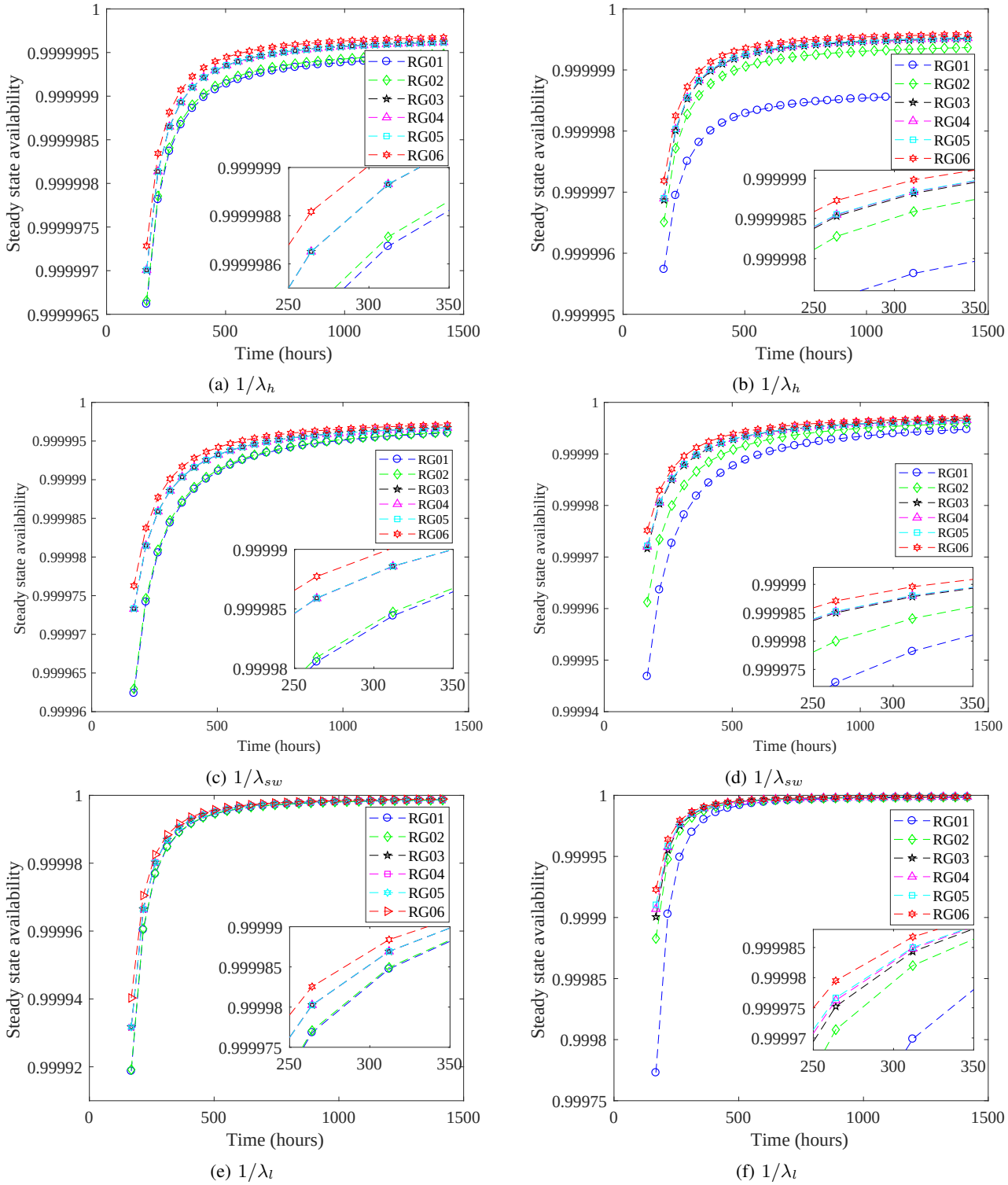
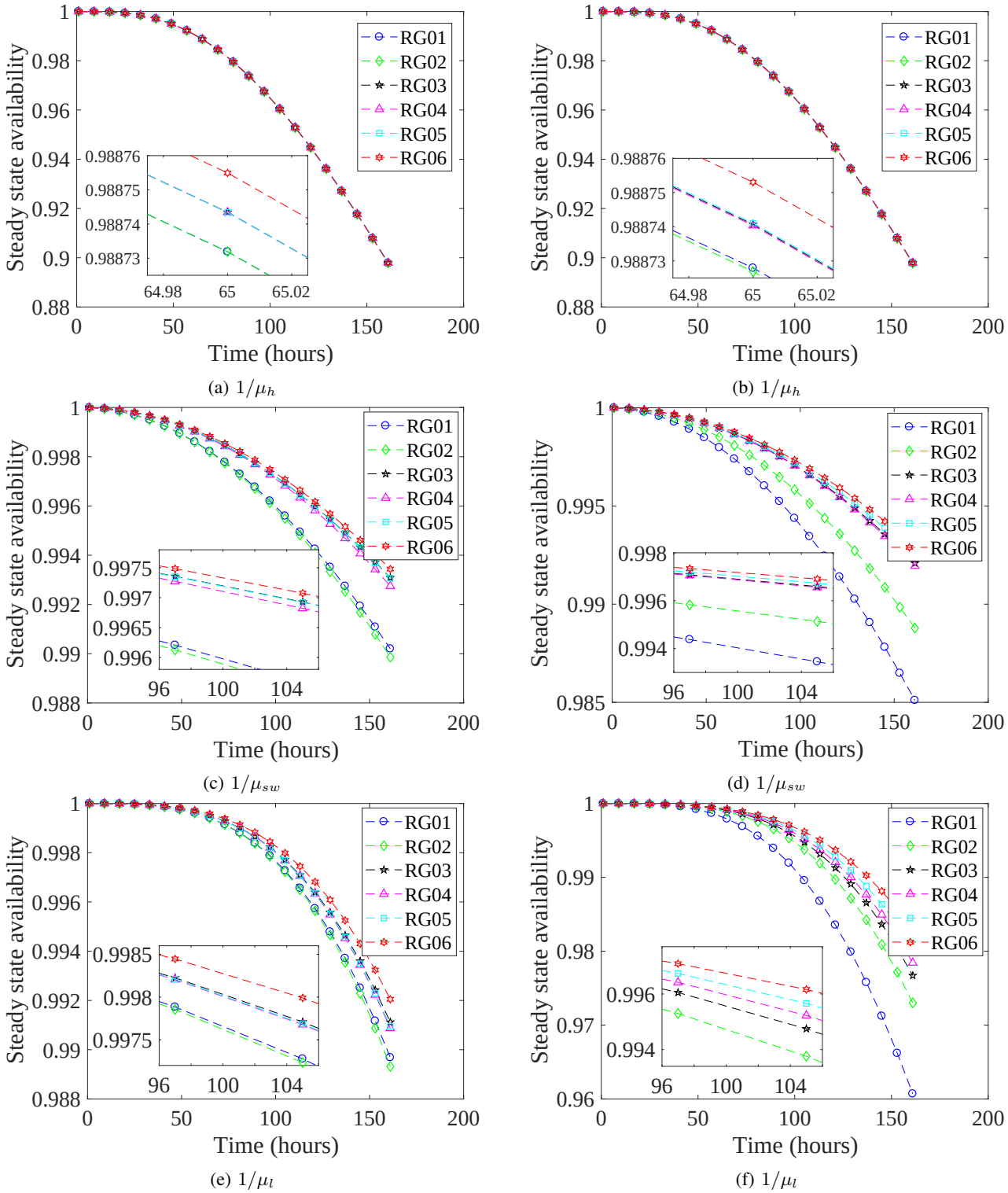


Figure 13: System Availability wrt. MTTFs in Case-studies
(a), (c), (e): Three-Tier; (b), (d), (f): Fat-Tree

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access



(a) $1/\mu_h$ (b) $1/\mu_h$

(c) $1/\mu_{sw}$ (d) $1/\mu_{sw}$

(e) $1/\mu_l$ (f) $1/\mu_l$

Figure 14: System Availability wrt. MTTRs in Case-studies
(a), (c), (e): Three-Tier; (b), (d), (f): Fat-Tree

C. LIMITATION AND DISCUSSION

Large DCN and dependencies

Our research focuses on developing a hierarchical modeling framework and applying the proposed framework to the analysis and evaluation of a particular type of DCN based on fat-tree topology. Modeling by using a hierarchical approach is very suitable when applied to computer networks in DCs. To investigate the correlation between the level of reliability/availability of the connection and routing topologies between compute nodes in DCN, we focus on applying the proposed hierarchical modeling framework to the modeling of tree-based switch-centric DCNs with a limited number of compute nodes and network devices. In practice, DCs often contain DCNs with a large number of physical servers, and these compute nodes also connect to many different peripherals to ensure data security, performance, monitoring and maintenance etc. (as shown in [94]).

Most of the analytical models in the area of dependability are likely challenged by the largeness and state-space explosion problems when considering large-scale systems. A not-well designed model with not correctly selected values of input parameters obviously causes an exponentially increasing of the number of states in solving the model, which eventually ends up with the impossibility to analyze the model. To avoid the state-space explosion of monolithic models, we develop a high level of abstraction in a hierarchical manner and appropriately set values of input parameters. However, for the large systems to be modeled using the proposed framework, we need to apply some approximate methods such as folding technique [95] and fixed-point iteration [96]. Ones can also develop interacting state-space models to study a large scale system as in [97, 98]. However, the use of such techniques often disregards the thorough consideration of overall network topologies and architectures of network elements as shown in this study. In some cases, ones even have to trade off between system architectures versus system behaviors rather than to hamornize the detailed incorporation of system architectures in modeling with the detailed incorporation of featured system behaviors while balancing the overall complexity of the system model.

In our study, we simplified the architecture and size of the system as well as the number of compute nodes in DCN for the sake of investigating DCN's service reliability/availability. Nevertheless, the proposed hierarchical modeling framework can be extended additionally in many respects to aim at reliability/availability evaluation of a large DCN. Thanks to the extensibility of RG, ones may scale up the analytical models for larger DCN architectures involving more number of network components. Extremely large models can be solved by computing reliability (availability) upper and lower bounds as shown in [51]. However, due to the repetition features of a DCN architecture, the large-scale DCN is not our main focus. We attempted to propose a specific hierarchical modeling framework and to explore the modeling capability of the framework which is not only able to capture complexity and flexibility of the network topologies at the top level but also able to incorporate sophisticated operations of system at the bottom level in a complete manner. This idea opens up a fruitful avenue for analyzing and evaluating large DCNs.

The most advantage of combinatorial models like RG and FT is that the models can quickly capture the overall architecture of the system while sacrificing the involvement of sophisticated behaviors such that of dependencies. The most disadvantage of state-space models like CTMC or SRN is that the models can detail at the lowest level of operational interactions and run-time dependencies but they likely encounter with state explosion problems or largeness problems when attempting to capture many sophisticated behaviors and interactions between the system components at once in a monolithic model of the same type or when attempting to cover multiple levels of the system architecture in one large model or when involving a number of heterogeneous components (for instance, a larger number of VMs running on a larger number of VMMs). For those reasons, the hierarchy of combinatorial models with integrated state-space models can help solve the issue when several assumptions of sophisticated dependencies and interactions are present to obtain the targeted measures of interest of the overall system. In this paper, we assume to not involve sophisticated dependencies and interactions between system components to simplify and reduce the size of the system model. However, the dependencies of VM on VMM and of VMM on physical hardwares in the proposed models were also taken into consideration in a simplified manner, in which we use MTTFeq and MTTReq of the lower-level subsystems to represent their uptime and downtime periods which affects the upper hosted subsystems. Further considerations could be refered in many of previous works [34, 35, 61, 76, 99]).

Routing topologies

The object studied in this paper is DCNs complying with the tree-based switch-centric topologies. However, as introduced in Section I, different topologies are increasingly deployed for DCNs in modern datacenters, that typically aim to optimize data transfer performance, reduce operational power consumption, or facilitate the management of resources. Because the advantage of modeling by using RG is that it can capture routing topologies or connections between components in the network with any architecture, the proposed hierarchical modeling framework can be used to model and evaluate other topologies of DCNs in literature. When reliablity and availability are the targeted metrics of interest, and the targeted system to be modeled has a multi-level complexity such as a network of systems or a system of systems, the modeling framework is helpful for different comprehensive assessment and analyses.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access

Evaluation metrics

The modeling framework in this study focuses on reliability/availability assessment, which provides significant indicators of high quality of services and constantly online business continuity for a computing system in DCs. The reliability/availability assessment relies on the consideration of failure modes and recovery actions in the system. The consideration of both performance and availability, which is roughly called perform-ability [100] is out of this study's scope. We do not consider the degradation of the system's performance in association with availability evaluation of the system. This extension to consider other evaluation metrics would be a fruitful topic and a broad avenue for future studies, but it may require proper modifications of the modeling framework. In some extreme cases, the physical computing system is not capable of providing sufficient resources to process a huge number of end-user requests or overloaded data traffics, resulting the consequence that the system is too busy to be unavailable to new requests even though the system is physically fault free. This is possibly perceived as a failure due to run-time operations of the system. The incorporation of system workload in modeling is usually in the cases when computing resources are the main concern or strict requirements for processing data-incentive tasks are associated with system design. Therefore, performace related metrics are considering in modeling and analysis often for specific resource consuming operations in specific systems such as data backup and store in a web service system of two servers [101]. The relation between reliability/availability and performance-related attributes in a certain system is an essential and fruitful topic but out of the scope of the study in this paper. When reliability/availability are the targeted metrics of interest, inherent failures and recovery actions related to the design of physical hardware and software subsystems are literally the main concern, such as a sudden failure of power supply system or aging failure of VM subsystem, which are statistically observed in long-term running. In these cases, the neglection of dynamic workloads impacting on reliability/availability is a necessary assumption, which is to assume that, the capacity of the computing resources in the DCN is expected to flexibly cover the variation of workloads either at highest peak or lowest level. So that, the peak amount of requests or high data traffic do not likely cause any severe system failure in run-time operation.

Due to a limited space, we limit ourself to select several impacting factors which also represent the main reliability/availability indicators of the main parts in the network including MTTFeq and MTTReq of physical servers, switches and links. Based on the output analyses of these parameters, ones may go further for detailed sensitivity analyses for additional requirements usually in industry with regard to other parameters that eventually constitute the above-mentioned main factors.

Practical implementation

One of the main advantages of the proposed hierarchical modeling framework is that we can take into account failure modes and recovery mechanisms as well as complex behaviors in the operations of DCNs from the lowest level of components to the overall level of a network system. However, increasing the complexity of the network of physical compute nodes or incorporating the different behaviors of each component in the system likely complicate the entire system model. This can lead to a largeness problem in analytical modeling of large networked systems. Nonetheless, one can use the techniques and algorithms proposed in [51] to reduce complexity in system modeling at RG models, or can apply typical solutions to avoid state-space explosion such as state truncation [102], state aggregation [103], model decomposition [104], state exploration [105, 106], and model composition [107, 108] in SRN modeling at components level. The attempt to predict different metrics of interest of a system by using analytical models is essentially to provide a reliable theoretical basis to facilitate system design processes, as well as to enhance system performance control processes in the long run. But, as a basic principle, the combination of theoretical results with the results obtained from a system simulation program, along with results of practical and experimental implementations produces the most reliable outcomes. For that reason, the comparison between theoretical and simulated results versus the results obtained from actual implementation in real-world is essential in future work.

VII. CONCLUSION

This paper presented a comprehensive hierarchical modeling and analysis of DCNs. The systems are based on tree-based switch-centric network topologies (three-tier and fat-tree), that consist of three layers of switching switches accompanying sixteen physical servers. We attempted to construct hierarchical models for the system consisting of three layers, including an RG at the system layer, a fault-tree at the subsystem layer, and SRN at the component layer. We also conducted a number of comprehensive analyses regarding reliability and availability. The results showed that the distribution of active nodes in the network can enhance the availability/reliability of cloud computing systems. Furthermore, the MTTF and MTTR of physical servers are the major impacting factors, whereas those of links are important in maintaining high availability for the system. The results of this study can facilitate the development and management of practical cloud computing centers.

ACKNOWLEDGMENT

This research was supported by the 2018 KU Brain Pool Program of Konkuk University, South Korea.

This research was partially supported by the MSIT(Ministry of Science, ICT), Korea, under the ITRC(Information Technology Research Center) sup-

port program(IITP-2018-2016-0-00465) supervised by the IITP(Institute for Information & communications Technology Promotion). This work was partially supported by Institute for Information and Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2017-0-00121 Development of a Traffic Predictive Simulation SW for Improving the Urban Traffic Congestion). This research was partly supported by the project CS19.10 managed by the Institute of Information Technology (IOIT). This research was financially supported by the Ministry of Trade, Industry and Energy (MOTIE) and Korean Institute for Advancement of Technology. (No. N0002431).

We would like to thank Prof. Dr. Kishor S. Trivedi and his research team at Duke Universiy, USA for providing the tool SHARPE. We also thank many peer-reviewers for their constructive and suggestive comments.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access

APPENDIX: GUARD AND REWARD FUNCTIONS IN SRN MODELS

Table 3: DEFINITION OF GUARD AND REWARD FUNCTIONS ATTACHED IN SRN MODELS

Name	Attached SRN Model/ Transition	Condition	
$ssa_{Upgrade}$	Switch: Upgrade	$1 : \#(P_{Unor}) > 0$	0: Otherwise
$ssa_{Chassis}$	Switch: Chassis	$1 : \#(P_{Cup}) > 0$	0: Otherwise
ssa_{LCin}	Switch: LC-in	$1 : \#(P_{LCiup}) > 0$	0: Otherwise
ssa_{LCout}	Switch: LC-out	$1 : \#(P_{LCoup}) > 0$	0: Otherwise
ssa_{CSC_SFC}	Switch: CSC_SFC	$1 : \#(P_{CSCup}) + \#(P_{SFCup}) >= 4$	0: Otherwise
ssa_{GRP}	Switch: GRP	$1 : \#(P_{GRPnor}) + \#(P_{GRPafd}) + \#(P_{GRPstd}) + \#(P_{GRPsfd}) + \#(P_{GRPsfu}) > 0$	0: Otherwise
ssa_{IOS}	Switch: IOS	$1 : \#(P_{IOSnor})+\#(P_{IOSafd})+\#(P_{IOSstd})+\#(P_{IOSsfd})+\#(P_{IOSsfu}) > 0$	0: Otherwise
ssa_{CPU}	Host: CPU	$1 : \#(P_{CPUnor}) >= m_{CPU}$	0: Otherwise
ssa_{MEM}	Host: MEM	$1 : \#(P_{MEMup}) >= m_{MEM}$	0: Otherwise
ssa_{NET}	Host: NET	$1 : \#(P_{NETup}) >= m_{NET}$	0: Otherwise
ssa_{PWR}	Host: PWR	$1 : \#(P_{PWRup}) >= m_{PWR}$	0: Otherwise
ssa_{COO}	Host: COO	$1 : \#(P_{COOup}) >= m_{COO}$	0: Otherwise
ssa_{VMM}	Host: VMM	$1 : \#(P_{VMMup}) + \#(P_{VMMfp}) > 0$	0: Otherwise
ssa_{VM}	Host: VM	$1 : \#(P_{VMup}) + \#(P_{VMfp}) > 0$	0: Otherwise
ssa_{APP}	Host: APP	$1 : \#(P_{APPup}) >= m_{APP}$	0: Otherwise
gT_{CPUf}	T_{CPUf}	$1 : \#(P_{CPUnor}) >= m_{CPU}$	0: Otherwise
$gT_{MEMdet1}$	$T_{MEMdet1}$	$1 : \#(P_{MEMdn}) < n_{MEM} - m_{MEM}$	0: Otherwise
$gT_{MEMdet2}$	$T_{MEMdet2}$	$1 : \#(P_{MEMdn}) >= n_{MEM} - m_{MEM}$	0: Otherwise
gT_{MEMf}	T_{MEMf}	$1 : \#(P_{MEMup}) >= m_{MEM}$	0: Otherwise
gT_{NETf}	T_{NETf}	$1 : \#(P_{NETup}) >= m_{NET}$	0: Otherwise
$gT_{NETdet1}$	$T_{NETdet1}$	$1 : \#(P_{NETdn}) < n_{NET} - m_{NET}$	0: Otherwise
$gT_{NETdet2}$	$T_{NETdet2}$	$1 : \#(P_{NETdn}) >= n_{NET} - m_{NET}$	0: Otherwise
gT_{PWRf}	T_{PWRf}	$1 : \#(P_{PWRup}) >= m_{PWR}$	0: Otherwise
$gT_{PWRdet1}$	$T_{PWRdet1}$	$1 : \#(P_{PWRdn}) < n_{PWR} - m_{PWR}$	0: Otherwise
$gT_{PWRdet2}$	$T_{PWRdet2}$	$1 : \#(P_{PWRdn}) >= n_{PWR} - m_{PWR}$	0: Otherwise
gT_{COOf}	T_{COOf}	$1 : \#(P_{COOup}) >= m_{COO}$	0: Otherwise
$gT_{COOdet1}$	$T_{COOdet1}$	$1 : \#(P_{COOdn}) < n_{COO} - m_{COO}$	0: Otherwise
$gT_{COOdet2}$	$T_{COOdet2}$	$1 : \#(P_{COOdn}) >= n_{COO} - m_{COO}$	0: Otherwise
$gt_{VMMuprej}$	$t_{VMMuprej}$	$1 : \#(P_{VMMclockreset}) == 1$	0: Otherwise
gT_{VMMclk}	T_{VMMclk}	$1 : \#(P_{VMMup}) + \#(P_{VMMfp}) > 0$	0: Otherwise
$gt_{VMMclkrej}$	$t_{VMMclkrej}$	$1 : \#(P_{VMMup}) + \#(P_{VMMfp}) > 0$	0: Otherwise
gt_{VMMupo}	t_{VMMupo}	$1 : \#(P_{VMMdup}) == 0$	0: Otherwise
gt_{VMMafo}	t_{VMMafo}	$1 : \#(P_{VMMdup}) == 0$	0: Otherwise
$gT_{VMMupin}$	$T_{VMMupin}$	$1 : (\#(P_{VMMdup}) == 1)\&\&(\#(P_{VMMup})+\#(P_{VMMfp})+\#(P_{VMMrej})+\#(P_{VMMaf})+\#(P_{VMMafd})+\#(P_{VMMf})+\#(P_{VMMfd}) < n_{VMM})$	0: Otherwise
$gt_{VMMclkreset}$	$t_{VMMclkreset}$	$1 : \#(P_{VMMrej}) == n_{VMM}$	0: Otherwise
$gt_{VMMafdo}$	$t_{VMMafdo}$	$1 : \#(P_{VMMdup}) == 0$	0: Otherwise
gt_{VMMfpo}	t_{VMMfpo}	$1 : \#(P_{VMMdup}) == 0$	0: Otherwise
gt_{VMMfo}	t_{VMMfo}	$1 : \#(P_{VMMdup}) == 0$	0: Otherwise
gt_{VMMfdo}	t_{VMMfdo}	$1 : \#(P_{VMMdup}) == 0$	0: Otherwise
$gt_{VMMfprej}$	$t_{VMMfprej}$	$1 : \#(P_{VMMclockreset}) == 1$	0: Otherwise
$gt_{VMMrejo}$	$t_{VMMrejo}$	$1 : \#(P_{VMMdup}) == 0$	0: Otherwise
gt_{VMafo}	t_{VMafo}	$1 : \#(P_{VMdup}) == 0$	0: Otherwise
gt_{VMupo}	t_{VMupo}	$1 : \#(P_{VMdup}) == 0$	0: Otherwise
gT_{VMupin}	T_{VMupin}	$1 : (\#(P_{VMdup}) == 1)\&\&(\#(P_{VMup}) + \#(P_{VMfp}) + \#(P_{VMrej}) + \#(P_{VMaf}) + \#(P_{VMafd}) + \#(P_{VMf}) + \#(P_{VMfd}) < n_{VM})$	0: Otherwise
$gt_{VMuprej}$	$t_{VMuprej}$	$1 : \#(P_{VMclockreset}) == 1$	0: Otherwise
$gt_{VMclkreset}$	$t_{VMclkreset}$	$1 : \#(P_{VMrej}) == n_{VM}$	0: Otherwise
gt_{VMrejo}	t_{VMrejo}	$1 : \#(P_{VMdup}) == 0$	0: Otherwise
$gt_{VMfprej}$	$t_{VMfprej}$	$1 : \#(P_{VMclockreset}) == 1$	0: Otherwise
gT_{VMclk}	T_{VMclk}	$1 : \#(P_{VMup}) + \#(P_{VMfp}) > 0$	0: Otherwise
gt_{VMafdo}	t_{VMafdo}	$1 : \#(P_{VMdup}) == 0$	0: Otherwise
gt_{VMfdo}	t_{VMfdo}	$1 : \#(P_{VMdup}) == 0$	0: Otherwise
gt_{VMfpo}	t_{VMfpo}	$1 : \#(P_{VMdup}) == 0$	0: Otherwise
gt_{VMfo}	t_{VMfo}	$1 : \#(P_{VMdup}) == 0$	0: Otherwise

Continued on next page

Continued from previous page

Name	Attached SRN Model/Transition	Condition	
$gt_{VMclkrej}$	$t_{VMclkrej}$	$1 : \#(P_{VMup}) + \#(P_{VMfp}) > 0$	0: Otherwise
$gt_{APPupdateo}$	$t_{APPupdateo}$	$1 : \#(P_{APPdup}) == 0$	0: Otherwise
$gT_{APPupin}$	$T_{APPupin}$	$1 : (\#(P_{APPdup}) == 1)\&\&(\#(P_{APPup}) + \#(P_{APPdn}) + \#(P_{APPupdate}) < n_{APP})$	0: Otherwise
gt_{APPupo}	t_{APPupo}	$1 : \#(P_{APPdup}) == 0$	0: Otherwise
gt_{APPdno}	t_{APPdno}	$1 : \#(P_{APPdup}) == 0$	0: Otherwise

Concluded

APPENDIX: STATES AND TRANSITIONS IN SRN MODELS OF HOST AND ROUTER

Table 4: DESCRIPTIONS OF STATES AND TRANSITIONS IN HOST SUB-MODELS

Name	Description
—CPU SRN Model—	
P_{CPUnor}, P_{CPUdn}	Operational and down states of a CPU, respectively
P_{CPUrp}	A CPU is in repair/re-installation processes.
T_{CPUf}	A CPU fails and enters a downtime period.
T_{CPUdet}, T_{CPUrp}	A repairperson is summonned to detect the failures of a CPU, and a CPU is repaired or re-installed, respectively.
—MEM, NET, PWR and COO SRN Models—	
P_{MEMup}, P_{NETup}, P_{PWRup}, P_{COOup}	Operational states of MEM, NET, PWR and COO subsystems, respectively
P_{MEMdn}, P_{NETdn}, P_{PWRdn}, P_{COOdn}	Downstates of MEM, NET, PWR and COO subsystems, respectively
P_{MEMrp1}, P_{NETrp1}, P_{PWRrp1}, P_{COOrp1}	The MEM, NET, PWR or COO subsystem is still running when one of its respective components is in repair/re-installation after failure detection.
P_{MEMrp2}, P_{NETrp2}, P_{PWRrp2}, P_{COOrp2}	A MEM, NET, PWR or COO component is in repair/re-installation after failure detection but the respective MEM, NET, PWR or COO subsystem entirely goes down.
T_{MEMf}, T_{NETf}, T_{PWRf}, T_{COOf}	A MEM, NET, PWR or COO fails, respectively.
$T_{MEMdet1}$, $T_{NETdet1}$, $T_{PWRdet1}$, $T_{COOdet1}$	A repairperson is detecting a failure of a MEM, NET, PWR or COO when the respective MEM, NET, PWR or COO subsystem is still running.
$T_{MEMdet2}$, $T_{NETdet2}$, $T_{PWRdet2}$, $T_{COOdet2}$	A repairperson is detecting a failure of a MEM, NET, PWR or COO when the respective MEM, NET, PWR or COO subsystem entirely goes down.
T_{MEMrp1}, T_{NETrp1}, T_{PWRrp1}, T_{COOrp1}	A MEM, NET, PWR or COO is repaired when the respective MEM, NET, PWR or COO subsystem is still operational.
T_{MEMrp2}, T_{NETrp2}, T_{PWRrp2}, T_{COOrp2}	A MEM, NET, PWR or COO is repaired when the respective MEM, NET, PWR or COO subsystem is in a downtime period.
—APP SRN Model—	
P_{APPup}, P_{APPdn}	An APP normally operates or stays in a downtime period, respectively.
$P_{APPupdate}$	An APP is in an update.
P_{APPdup}, P_{APPddn}	Underlying layer of the APP subsystem is operational or in a downtime period, respectively.
T_{APPf}, T_{APPr}	An APP fails or is restarted, respectively.
T_{APPudr}	An APP enters an update process.
$T_{APPupdate}$, T_{APPupi}	An APP has been updated, or an APP is started and runs in normal state, respectively.
T_{APPdf}, T_{APPdr}	The underlying layer of an APP subsystem enters a downtime period and recovery process, respectively.
t_{APPupo}	An APP in running state is suddenly aborted when the underlying subsystems enter downtime.
t_{APPdno}	An APP in failure state (with temporary stored memory) is suddendly vanished when the underlying subsystems enter downtime.
$t_{APPupdateo}$	An APP in updating process is suddenly aborted when the underlying subsystems enter downtime.
—VMM, VM SRN Models—	
P_{VMMup}, P_{VMup}	Normal states of VMM and VM, respectively
P_{VMMf}, P_{VMf}	Respective failure states of a VMM, VM
P_{VMMfd}, P_{VMfd}	Failure detection states in which a VMM or VM undergoes a failure detection process, respectively.
P_{VMMfp}, P_{VMfp}	Failure-probable states in which a VMM or VM may fail with high probability, but still in operation.
P_{VMMrej}, P_{VMrej}	A VMM or VM is in a software rejuvenation process.
P_{VMMaf}, P_{VMaf}	States in which a VMM or VM fails due to aging without rejuvenation.
P_{VMMafd}, P_{VMafd}	States in which failures due to aging of VMM/VM are in a detection process.
P_{VMMclk}, P_{VMclk}	Counting-time state of a VMM, VM clock
$P_{VMMclktrig}$, $P_{VMclktrig}$	State of VMM, VM clocks in which the clock's couting time elements are accumulated to trigger software rejuvenation on a VMM or VM.
$P_{VMMrejtrig}$, $P_{VMrejtrig}$	State of VMM, VM clocks when software rejuvenation processes are triggered on a VMM or VM.
T_{VMMupi}, T_{VMupi}	Create a new VMM or VM and set operational states of the created ones.
T_{VMMf}, T_{VMf}	A VMM or VM fails not because of aging.

Continued on next page

Continued from previous page

Name	Description
T_{VMMfd}, T_{VMfd}	A normal failure of a VMM or VM is under detection.
T_{VMMfr}, T_{VMfr}	A failed VMM or VM is under repair.
T_{VMMfp}, T_{VMfp}	A VMM or VM transits from a normal operational state to a failure-probable state due to aging problems.
T_{VMMaf}, T_{VMaf}	A VMM or VM enters an aging-failure period because of aging faults.
T_{VMMafd}, T_{VMafd}	Aging-failures of a VMM or VM are detected.
T_{VMMafr}, T_{VMafr}	A repair of aging-failures is conducted for a VMM/VM.
T_{VMMrej}, T_{VMrej}	A VMM or VM undergoes a software rejuvenation process after periodically triggered by the VMM/VM's clocks.
$T_{VMMclki}, T_{VMclki}$	Creation of a VMM/VM subsystems' clock couting elements.
t_{VMMupo}, t_{VMupo}	Removal of a VMM/VM in a normal state if there is a failure of its underlying subsystem.
t_{VMMfo}, t_{VMfo}	Removal of a failed VMM/VM after a failure of its underlying subsystem.
t_{VMMfdo}, t_{VMfdo}	Removal of a VMM/VM in detection process after a failure of its underlying subsystem.
t_{VMMfpo}, t_{VMfpo}	Removal of a VMM/VM in failure-probable state due to a failure of its underlying subsystem.
t_{VMMafo}, t_{VMafo}	Removal of a VMM/VM in aging-failure state after a failure of its underlying subsystem.
$t_{VMMafdo}, t_{VMafdo}$	Removal of a VMM/VM in detection of aging-failure if there is a failure of its underlying subsystem.
$t_{VMMuprej}, t_{VMuprej}$	A software rejuvenation process is triggered on a VMM/VM running in normal operational state.
$t_{VMMfprej}, t_{VMfprej}$	A software rejuvenation process is triggered on a VMM/VM running in failure-probable state.
$t_{VMMrejo}, t_{VMrejo}$	A VMM/VM in rejuvenation process is removed due to a failure of its underlying subsystem.

Concluded

Table 5: DESCRIPTIONS OF STATES AND TRANSITIONS IN ROUTER SUB-MODELS

Name	Description
—Upgrade SRN Model—	
P_{Unor}, P_{Uup}	Normal running and upgrading states of a switch, respectively.
T_{Run} , $T_{Upgrade}$	A switch transits to a ready-to-upgrate state after running in a normal state for a certain period of time; and an upgrade process is conducted on a switch.
—Chassis SRN Model—	
P_{Cup}, P_{Cdn}	The chassis of the switch is in normal state or is in malfunction, respectively.
T_{Cf}, T_{Cr}	The chassis fails to work properly or it is repaired or replaced.
—LC-in, LC-out SRN Models—	
P_{LCiup}, P_{LCoup}	Operational states of LC-in and LC-out, respectively
P_{LCihd}, P_{LCohd}	Down-states of hardware modules in LC-in and LC-out, respectively
P_{LCisd}, P_{LCosd}	Down-states of software modules in LC-in and LC-out, respectively
T_{LCihf}, T_{LCohf}	Hardware modules of LC-in and LC-out fail, respectively.
T_{LCisf}, T_{LCosf}	Software modules of LC-in and LC-out fail, respectively.
T_{LCihr}, T_{LCohr}	Hardware modules of LC-in and LC-out are repaired, respectively.
T_{LCisr}, T_{LCosr}	Software modules of LC-in and LC-out are repaired, respectively.
—CSC-SFC SRN Model—	
P_{CSCup}, P_{SFCup}	Normal states of CSC and SFC, respectively
P_{CSCdn}, P_{SFCdn}	Down states of CSC and SFC, respectively
T_{CSCf}, T_{SFCf}	A CSC and a SFC fails, respectively
T_{CSCr}, T_{SFCr}	A CSC and a SFC is repaired, respectively
—GRP, IOS SRN Model—	
P_{GRPnor}, P_{IOSnor}	All hardwares of GRP, IOS are working properly in normal states, respectively.
P_{GRPafu}, P_{IOSafu}	The states of GRP and IOS respectively, in which there is a failure of active units and the units are not able to switch to standby.
P_{GRPsfu}, P_{IOSsfu}	Failure and undetected states of standby units in GRP and IOS, respectively.
P_{GRPslf}, P_{IOSslf}	GRP or IOS are running on switched standby units with laten failure, respectively.
P_{GRPafd}, P_{IOSafd}	GRP or IOS has one of the active unit in a failure state but it is detected.
P_{GRPstd}, P_{IOSstd}	GRP or IOS runs on a standby unit which took over the previously failed active unit, respectively.
P_{GRPsfd}, P_{IOSsfd}	There is a failed standby unit but detected in GRP or IOS, respectively.
P_{GRPa2f}, P_{IOSa2f}	Second unit in GRP or IOS failed while trying to recover, respectively.
T_{GRPafu}, T_{IOSafu}	Active units failed and could not switch to standby ones in GRP and IOS, respectively.
T_{GRPafr}, T_{IOSafr}	Faults of the previous active units are removed, those units return to normal state in GRP and IOS, respectively.
T_{GRPsfu}, T_{IOSsfu}	Standby units failed and the failures are undetected in GRP and IOS, respectively.
T_{GRPslf}, T_{IOSslf}	The GRP or IOS switched to standby units with latent failure, respectively.
$T_{GRPslfr}, T_{IOSslfr}$	The latent faults of standby units in GRP and IOS are repaired, respectively.
T_{GRPafd}, T_{IOSafd}	One of the active units is failed and detected in GRP or IOS, respectively.
T_{GRPstd}, T_{IOSstd}	The standby units in GRP or IOS has taken over the previously failed active unit, respectively.
$T_{GRPafdr}, T_{IOSafdr}$	The previously failed active unit is repaired in GRP or IOS, respectively.
T_{GRPa2f}, T_{IOSa2f}	All the active and standby units in GRP or IOS respectively, transit a failure state.
T_{GRPsfd}, T_{IOSsfd}	A standby unit in GRP or IOS fails, and is detected, respectively.

Continued on next page

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2891282, IEEE Access

Continued from previous page

Name	Description
$T_{GRPsfdr}, T_{IOSsfdr}$	The previously failed standby unit in GRP or IOS is repaired, respectively.
$T_{GRPsfd.a2f}, T_{IOSsfd.a2f}$	The active unit fails while trying to recover the failed standby.
$T_{GRPa2fr}, T_{IOSa2fr}$	Two failed units (active and standby) are recovered and returned to normal state.

Concluded

References

[1] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, and M. F. Zhani, "Data Center Network Virtualization: A Survey", *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 909–928, 2013, ISSN: 1553-877X. DOI: 10.1109/SURV.2012.090512.00043 (cit. on p. 1).

[2] R. Cocchiara, H. Davis, and D. Kinnaird, "Data center topologies for mission-critical business systems", *IBM Systems Journal*, vol. 47, no. 4, pp. 695–706, 2008, ISSN: 0018-8670. DOI: 10.1147/SJ.2008.5386510 (cit. on p. 1).

[3] T. Chen, X. Gao, and G. Chen, "The features, hardware, and architectures of data center networks: A survey", *Journal of Parallel and Distributed Computing*, vol. 96, pp. 45–74, 2016, ISSN: 07437315. DOI: 10.1016/j.jpdc.2016.05.009 (cit. on p. 2).

[4] S. Zafar, A. Bashir, and S. A. Chaudhry, "On implementation of DCTCP on three-tier and fat-tree data center network topologies", *SpringerPlus*, vol. 5, no. 1, p. 766, 2016, ISSN: 2193-1801. DOI: 10.1186/s40064-016-2454-4 (cit. on p. 2).

[5] M. Al-Fares, A. Loukissas, and A. Vahdat, *A scalable, commodity data center network architecture*, 2008. DOI: 10.1145/1402946.1402967 (cit. on pp. 2–3).

[6] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "PortLand: a scalable fault-tolerant layer 2 data center network fabric", in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication (SIGCOMM '09)*, ser. SIGCOMM '09, ACM, 2009, pp. 39–50 (cit. on p. 2).

[7] G. Chen, Y. Zhao, D. Pei, and D. Li, "Rewiring 2 Links Is Enough: Accelerating Failure Recovery in Production Data Center Networks", in *2015 IEEE 35th International Conference on Distributed Computing Systems*, IEEE, 2015, pp. 569–578, ISBN: 978-1-4673-7214-5. DOI: 10.1109/ICDCS.2015.64 (cit. on p. 2).

[8] Y. Liu, D. Lin, J. Muppala, and M. Hamdi, "A study of fault-tolerance characteristics of data center networks", English, in *IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN 2012)*, IEEE, 2012, pp. 1–6, ISBN: 978-1-4673-2266-9. DOI: 10.1109/DSNW.2012.6264696 (cit. on pp. 2–3).

[9] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell", in *Proceedings of the ACM SIGCOMM 2008 conference on Data communication - SIGCOMM '08*, vol. 38, New York, New York, USA: ACM Press, 2008, p. 75, ISBN: 9781605581750. DOI: 10.1145/1402958.1402968 (cit. on pp. 2–3).

[10] 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 2009)*, IEEE, 2009, pp. 2276–2285, ISBN: 978-1-4244-3512-8. DOI: 10.1109/INFCOM.2009.5062153 (cit. on p. 2).

[11] Cong Wang, Cuirong Wang, Ying Yuan, and Yongtao Wei, "MCube: A high performance and fault-tolerant network architecture for data centers", in *2010 International Conference On Computer Design and Applications*, IEEE, 2010, pp. V5–423–V5–427, ISBN: 978-1-4244-7164-5. DOI: 10.1109/ICCDA.2010.5540940 (cit. on p. 2).

[12] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: a hybrid electrical/optical switch architecture for modular data centers", *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, p. 339, 2010, ISSN: 01464833. DOI: 10.1145/1851275.1851223 (cit. on p. 2).

[13] H. M. Helal and R. E. Ahmed, "Performance evaluation of datacenter network topologies with link failures", in *2017 7th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO)*, IEEE, 2017, pp. 1–5, ISBN: 978-1-5090-5454-1. DOI: 10.1109/ICMSAO.2017.7934898 (cit. on p. 2).

[14] N. Farrington and A. Andreyev, "Facebook's data center network architecture", in *2013 Optical Interconnects Conference*, IEEE, 2013, pp. 49–50, ISBN: 978-1-4673-5063-1. DOI: 10.1109/OIC.2013.6552917 (cit. on p. 2).

[15] B. Lebiednik, A. Mangal, and N. Tiwari, "A Survey and Evaluation of Data Center Network Topologies", *CoRR*, vol. abs/1605.0, 2016. arXiv: 1605.01701 (cit. on p. 2).

[16] F. Yao, J. Wu, G. Venkataramani, and S. Subramaniam, "A comparative analysis of data center network architectures", in *2014 IEEE International Conference on Communications (ICC)*, IEEE, 2014, pp. 3106–3111, ISBN: 978-1-4799-2003-7. DOI: 10.1109/ICC.2014.6883798 (cit. on p. 2).

[17] Ponemon Institute, "2013 Cost of Data Center Outages", Emerson Network Power, Tech. Rep. (cit. on p. 2).

[18] R. Miller, "Failure Rates in Google Data Centers (Report)", Data Center Knowledge, Tech. Rep., 2008 (cit. on p. 2).

[19] T. Lumpp, J. Schneider, J. Holtz, M. Mueller, N. Lenz, a. Biazetti, and D. Petersen, "From high availability and disaster recovery to business continuity solutions", *IBM Systems Journal*, vol. 47, no. 4, pp. 605–619, 2008, ISSN: 0018-8670. DOI: 10.1147/SJ.2008.5386516 (cit. on pp. 2, 5).

[20] D. M. Gomes, P. T. Endo, G. Goncalves, D. Rosendo, G. L. Santos, J. Kelner, D. Sadok, and M. Mahloo, "Evaluating the cooling subsystem availability on

a Cloud data center", in *2017 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, 2017, pp. 736–741, ISBN: 978-1-5386-1629-1. DOI: 10.1109/ISCC.2017.8024615 (cit. on p. 2).

[21] T. A. Nguyen, D. S. Kim, and J. S. Park, "Availability Modeling and Analysis of Data Center Systems using Stochastic Reward Nets", Ph.D. Dissertation, Korea Aerospace University (KAU), 2015, p. 177 (cit. on pp. 2, 4, 6).

[22] F. Machida, J. Xiang, K. Tadano, and Y. Maeno, "Composing hierarchical stochastic model from SysML for system availability analysis", in *2013 IEEE 24th International Symposium on Software Reliability Engineering, ISSRE 2013*, 2013, pp. 51–60, ISBN: 9781479923663. DOI: 10.1109/ISSRE.2013.6698904 (cit. on p. 3).

[23] K. Bilal, M. Manzano, S. U. Khan, E. Calle, and A. Y. Zomaya, "On the Characterization of the Structural Robustness of Data Center Networks", *IEEE Transactions on Cloud Computing*, vol. 1, no. 1, pp. 1–1, 2013, ISSN: 2168-7161. DOI: 10.1109/TCC.2013.6 (cit. on pp. 3–4).

[24] 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, ISSN: 1089-7798. DOI: 10.1109/LCOMM.2013.091913.131176 (cit. on p. 3).

[25] R. d. S. Couto, S. Secci, M. E. M. Campista, and L. H.M. K. Costa, "Reliability and Survivability Analysis of Data Center Network Topologies", *Journal of Network and Systems Management*, vol. 24, no. 2, pp. 346–392, 2016, ISSN: 1064-7570. DOI: 10.1007/s10922-015-9354-8. arXiv: 1510.02735 (cit. on p. 3).

[26] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring Network Structure, Dynamics, and Function using NetworkX", in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11–15 (cit. on p. 3).

[27] 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, ISSN: 15241904. DOI: 10.1002/asmb.951 (cit. on pp. 3, 23–24).

[28] 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*, New York, New York, USA: ACM Press, 2014, pp. 75–82, ISBN: 9781450327947. DOI: 10.1145/2601381.2601389 (cit. on pp. 3–4).

[29] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications", in *Proceedings of ACM SIGCOMM*, 2011, ISBN: 978-1-4503-0797-0. DOI: 10.1145/2018436.2018477 (cit. on p. 3).

[30] P. Patel, A. Ranabahu, and A. Sheth, "Service Level Agreement in Cloud Computing", Tech. Rep., 2009 (cit. on p. 4).

[31] 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, ISSN: 0018-8670. DOI: 10.1147/SJ.2008.5386524 (cit. on pp. 4, 6–7, 13, 21).

[32] S. Distefano, F. Longo, M. Scarpa, and K. S. Trivedi, "Non-Markovian Modeling of a BladeCenter Chassis Midplane", in, Springer International Publishing, 2014, pp. 255–269. DOI: 10.1007/978-3-319-10885-8_18 (cit. on p. 4).

[33] G. Callou, J. Ferreira, P. Maciel, D. Tutsch, and R. Souza, "An Integrated Modeling Approach to Evaluate and Optimize Data Center Sustainability, Dependability and Cost", en, *Energies*, vol. 7, no. 1, pp. 238–277, 2014, ISSN: 1996-1073. DOI: 10.3390/en7010238 (cit. on p. 4).

[34] D. S. Kim, F. Machida, and K. S. Trivedi, "Availability Modeling and Analysis of a Virtualized System", in *2009 15th IEEE Pacific Rim International Symposium on Dependable Computing*, vol. 1, IEEE, 2009, pp. 365–371, ISBN: 978-0-7695-3849-5. DOI: 10.1109/PRDC.2009.64 (cit. on pp. 4, 6–7, 37).

[35] T. A. Nguyen, D. Min, and E. Choi, "A comprehensive evaluation of availability and operational cost for a virtualized server system using stochastic reward nets", *The Journal of Supercomputing*, pp. 1–55, 2017, ISSN: 1573-0484. DOI: 10.1007/s11227-017-2127-2 (cit. on pp. 4, 6, 13, 37).

[36] C. Melo, J. Dantas, J. Araujo, P. Maciel, R. Branchini, and L. Kawakami, "Availability models for synchronization server infrastructure", in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2016, pp. 003 658–003 663, ISBN: 978-1-5090-1897-0. DOI: 10.1109/SMC.2016.7844802 (cit. on p. 4).

[37] J. Dantas, R. Matos, J. Araujo, and P. Maciel, "An availability model for eucalyptus platform: An analysis of warm-standy replication mechanism", English, in *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2012, pp. 1664–1669, ISBN: 978-1-4673-1714-6. DOI: 10.1109/ICSMC.2012.6377976 (cit. on p. 4).

[38] R. Matos, J. Dantas, J. Araujo, K. S. Trivedi, and P. Maciel, "Redundant Eucalyptus Private Clouds: Availability Modeling and Sensitivity Analysis", *Journal of Grid Computing*, pp. 1–22, 2016, ISSN: 1570-7873. DOI: 10.1007/s10723-016-9381-z (cit. on p. 4).

[39] I. Costa, J. Araujo, J. Dantas, E. Campos, F. A. Silva, and P. Maciel, "Availability Evaluation and Sensitivity Analysis of a Mobile Backend-as-a-service

Platform", *Quality and Reliability Engineering International*, n/a–n/a, 2015, ISSN: 07488017. DOI: 10.1002/qre.1927 (cit. on p. 4).

[40] W. Hou, Z. Ning, L. Guo, and M. S. Obaidat, "Service Degradability Supported by Forecasting System in Optical Data Center Networks", *IEEE Systems Journal*, pp. 1–12, 2018, ISSN: 1932-8184. DOI: 10.1109/JSYST.2018.2821714 (cit. on p. 4).

[41] J. Yao, P. Lu, L. Gong, and Z. Zhu, "On Fast and Coordinated Data Backup in Geo-Distributed Optical Inter-Datacenter Networks", *Journal of Lightwave Technology*, pp. 1–1, 2015, ISSN: 0733-8724. DOI: 10.1109/JLT.2015.2425303 (cit. on p. 4).

[42] T. Wang, Z. Su, Y. Xia, B. Qin, and M. Hamdi, "Towards cost-effective and low latency data center network architecture", *Computer Communications*, 2016, ISSN: 01403664. DOI: 10.1016/j.comcom.2016.02.016 (cit. on p. 4).

[43] H.-P. Jiang, D. Chuck, and W.-M. Chen, "Energy-Aware Data Center Networks", *Journal of Network and Computer Applications*, 2016, ISSN: 10848045. DOI: 10.1016/j.jnca.2016.04.003 (cit. on p. 4).

[44] R. S. Couto, M. E. M. Campista, and L. H.M. K. Costa, "A reliability analysis of datacenter topologies", in *2012 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2012, pp. 1890–1895, ISBN: 978-1-4673-0921-9. DOI: 10.1109/GLOCOM.2012.6503391 (cit. on p. 4).

[45] T. A. Nguyen, D. Min, and J. S. Park, "A Comprehensive Sensitivity Analysis of a Data Center Network with Server Virtualization for Business Continuity", *Mathematical Problems in Engineering*, vol. 2015, pp. 1–20, 2015, ISSN: 1024-123X. DOI: 10.1155/2015/521289 (cit. on p. 4).

[46] T. A. Nguyen, T. Eom, S. An, J. S. Park, J. B. Hong, and D. S. Kim, "Availability Modeling and Analysis for Software Defined Networks", in *2015 IEEE 21st Pacific Rim International Symposium on Dependable Computing (PRDC)*, Zhangjiajie, China: IEEE, 2015, pp. 159–168, ISBN: 978-1-4673-9376-8. DOI: 10.1109/PRDC.2015.27 (cit. on p. 5).

[47] K. S. Trivedi and A. Bobbio, *Reliability and Availability Engineering: Modeling, Analysis, and Applications*, 1st. Cambridge: Cambridge University Press, 2017, p. 730, ISBN: 9781316163047. DOI: 10.1017/9781316163047 (cit. on pp. 5, 11–13, 23, 25).

[48] D. Bailey, E. Frank-Schultz, P. Lindeque, and J. L. Temple, III, "Three reliability engineering techniques and their application to evaluating the availability of IT systems: An introduction", *IBM Systems Journal*, vol. 47, no. 4, pp. 577–589, 2008, ISSN: 0018-8670. DOI: 10.1147/SJ.2008.5386507 (cit. on p. 5).

[49] C. J. Colbourn, *The Combinatorics of Network Reliability*. New York, USA: Oxford University Press, Inc., 1987, p. 176, ISBN: 0195049209 (cit. on p. 5).

[50] S. Sebastio, K. S. Trivedi, D. Wang, and X. Yin, "Fast computation of bounds for two-terminal network reliability", *European Journal of Operational Research*, vol. 238, no. 3, pp. 810–823, 2014, ISSN: 03772217. DOI: 10.1016/j.ejor.2014.04.035 (cit. on p. 5).

[51] A. V. Ramesh, K. S. Trivedi, T. C. Sharma, D Wang, D. W. Twigg, L. P. Nguyen, and Y Liu, *Reliability estimation methods for large networked systems*, 2014 (cit. on pp. 5, 7, 37–38).

[52] T. Thein, S.-D. Chi, and J. S. Park, "Availability Modeling and Analysis on Virtualized Clustering with Rejuvenation", *IJCSNS International Journal of Computer Science and Network Security*, vol. 8, no. 9, pp. 72–80, 2008 (cit. on p. 5).

[53] T. Thein and J. S. Park, "Availability Analysis of Application Servers Using Software Rejuvenation and Virtualization", *Journal of Computer Science and Technology*, vol. 24, no. 2, pp. 339–346, 2009, ISSN: 1000-9000. DOI: 10.1007/s11390-009-9228-1 (cit. on p. 5).

[54] R. D. S. Matos, P. R. M. Maciel, F. Machida, D. S. Kim, and K. S. Trivedi, "Sensitivity Analysis of Server Virtualized System Availability", *IEEE Transactions on Reliability*, vol. 61, no. 4, pp. 994–1006, 2012, ISSN: 0018-9529. DOI: 10.1109/TR.2012.2220711 (cit. on p. 6).

[55] T. T. Lwin and T. Thein, "High Availability Cluster System for Local Disaster Recovery with Markov Modeling Approach", *International Journal of Computer Science Issues, IJCSI*, vol. 6, no. 2, pp. 25–32, 2009. arXiv: 0912.1835 (cit. on p. 6).

[56] K. Trivedi, "Analysis of periodic preventive maintenance with general system failure distribution", in *Proceedings 2001 Pacific Rim International Symposium on Dependable Computing*, IEEE Comput. Soc, 2001, pp. 103–107, ISBN: 0-7695-1414-6. DOI: 10.1109/PRDC.2001.992686 (cit. on p. 6).

[57] F. Bause, P. S. Kritzinger, and D. Network, "Stochastic Petri Nets", pp. 133–140, 2002. DOI: 10.1007/978-3-322-86501-4_8 (cit. on p. 6).

[58] G. Ciardo, J. Muppala, and K. S. Trivedi, "SPNP: stochastic Petri net package", *Proceedings of the Third International Workshop on Petri Nets and Performance Models, PNPM89*, pp. 142–151, 1989. DOI: 10.1109/PNPM.1989.68548 (cit. on pp. 6, 25).

[59] G. Ciardo, J. K. Muppala, and K. S. Trivedi, "Analyzing concurrent and fault-tolerant software using stochastic reward nets", *Journal of Parallel and Distributed Computing*, vol. 15, no. 3, pp. 255–269, 1992, ISSN: 07437315. DOI: 10.1016/0743-7315(92)90007-A (cit. on p. 6).

[60] K. Han, T. A. Nguyen, D. Min, and E. M. Choi, "An Evaluation of Availability, Reliability and Power Consumption for a SDN Infrastructure Using Stochastic Reward Net", in *Advances in Computer Science and*

Ubiquitous Computing: CSA-CUTE2016, J. J.J. H. Park, Y. Pan, G. Yi, and V. Loia, Eds., Singapore: Springer Singapore, 2017, pp. 637–648, ISBN: 978-981-10-3023-9. DOI: 10.1007/978-981-10-3023-9_98 (cit. on p. 6).

[61] 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, ISSN: 01665316. DOI: 10.1016/j.peva.2012.09.003 (cit. on pp. 6, 22, 37).

[62] X. Yin, J. Alonso, F. Machida, E. C. Andrade, and K. S. Trivedi, "Availability Modeling and Analysis for Data Backup and Restore Operations", in *2012 IEEE 31st Symposium on Reliable Distributed Systems*, IEEE, 2012, pp. 141–150, ISBN: 978-1-4673-2397-0. DOI: 10.1109/SRDS.2012.9 (cit. on p. 6).

[63] M. Torquato, I. M. Umesh, and P. Maciel, "Models for availability and power consumption evaluation of a private cloud with VMM rejuvenation enabled by VM Live Migration", *The Journal of Supercomputing*, 2018, ISSN: 0920-8542. DOI: 10.1007/s11227-018-2485-4 (cit. on p. 6).

[64] T. A. Nguyen, D. S. Kim, and J. S. Park, "Availability modeling and analysis of a data center for disaster tolerance", *Future Generation Computer Systems*, vol. 56, pp. 27–50, 2016, ISSN: 0167739X. DOI: 10.1016/j.future.2015.08.017 (cit. on p. 6).

[65] T. A. Nguyen, D. Min, and E. Choi, "Stochastic Reward Net-based Modeling Approach for Availability Quantification of Data Center Systems", in *Dependability Engineering*, F. P. García Márquez, Ed., Rijeka: InTech, 2018, ch. 5. DOI: 10.5772/intechopen.74306 (cit. on pp. 6, 13).

[66] K. S. Trivedi, D. S. Kim, and X. Yin, "Multi-State Availability Modeling in Practice", in, 2012, pp. 165–180. DOI: 10.1007/978-1-4471-2207-4_12 (cit. on p. 6).

[67] R. Sahner, K. S. Trivedi, and A. Puliafito, "Hierarchical Models", in *Performance and Reliability Analysis of Computer Systems*, Boston, MA: Springer US, 1996, pp. 261–311. DOI: 10.1007/978-1-4615-2367-3_11 (cit. on p. 6).

[68] V. Lira, E. Tavares, and P. Maciel, "An automated approach to dependability evaluation of virtual networks", *Computer Networks*, 2015, ISSN: 13891286. DOI: 10.1016/j.comnet.2015.05.016 (cit. on p. 6).

[69] B. Silva, G. Callou, E. Tavares, P. Maciel, J. Figueiredo, E. Sousa, C. Araujo, F. Magnani, and F. Neves, "ASTRO: An integrated environment for dependability and sustainability evaluation", *Sustainable Computing: Informatics and Systems*, vol. 3, no. 1, pp. 1–17, 2013, ISSN: 22105379. DOI: 10.1016/j.suscom.2012.10.004 (cit. on p. 6).

[70] K. S.T.K. S. Trivede, R. V. R. Vasireddy, D. T. D. Trindale, S. N. S. Nathan, R. C. R. Castro, K. S. Trivedi, R. V. R. Vasireddy, D. T. D. Trindale, S. N. S. Nathan, and R. C. R. Castro, "Modeling High Availability Systems", English, in *2006 12th Pacific Rim International Symposium on Dependable Computing (PRDC'06)*, IEEE, 2006, pp. 154–164, ISBN: 0-7695-2724-8. DOI: 10.1109/PRDC.2006.45 (cit. on p. 6).

[71] M. Malhotra and K. Trivedi, "Power-hierarchy of dependability-model types", *IEEE Transactions on Reliability*, vol. 43, no. 3, pp. 493–502, 1994, ISSN: 00189529. DOI: 10.1109/24.326452 (cit. on pp. 6–7).

[72] R. Matos, A. Guimaraes, K. Camboim, P. Maciel, and K. Trivedi, "Sensitivity analysis of availability of redundancy in computer networks", in *CTRQ 2011, The Fourth International Conference on Communication Theory, Reliability, and Quality of Service*, 2011, pp. 115–121, ISBN: 978-1-61208-126-7 (cit. on p. 8).

[73] K. S. Trivedi, D. S. Kim, A. Roy, and D. Medhi, "Dependability and security models", *Proceedings of the 2009 7th International Workshop on the Design of Reliable Communication Networks, DRCN 2009*, pp. 11–20, 2009. DOI: 10.1109/DRCN.2009.5340029 (cit. on pp. 10, 23).

[74] K. S. Trivedi, S. Hunter, S. Garg, and R. Fricks, "Reliability Analysis Techniques Explored Through a Communication Network Example", North Carolina State University. Center for Advanced Computing and Communication, Tech. Rep. (cit. on p. 10).

[75] a. Addaim, a. Kherras, and B. Zantou, "Design of Store and Forward Data Collection Low-cost Nanosatellite", *2007 IEEE Aerospace Conference*, pp. 1–10, 2007. DOI: 10.1109/AERO.2007.353099 (cit. on p. 11).

[76] 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, pp. 1–18, 2014, ISSN: 2356-6140. DOI: 10.1155/2014/165316 (cit. on pp. 13, 22, 25, 37).

[77] T. A. Nguyen, D. Min, and Eun Mi Choi, "A Comprehensive Evaluation of Availability and Operation Cost for a Virtualized Servers System using Stochastic Reward Nets", Konkuk University, Seoul, Korea, 2016 (cit. on p. 13).

[78] T. A. Nguyen, K. Han, D. Min, E. Choi, T. D. Thang, and Y.-J. Choi, "A stochastic reward net-based assessment of reliability, availability and operational cost for a software-defined network infrastructure", *The Journal of Supercomputing*, 2018, ISSN: 0920-8542. DOI: 10.1007/s11227-018-2677-y (cit. on p. 13).

[79] F. Machida, R. Xia, and K. Trivedi, "Performability Modeling for RAID Storage Systems by Markov Regenerative Process", *IEEE Transactions on De-*

pendable and Secure Computing, vol. PP, no. 99, pp. 1–1, 2015, ISSN: 1545-5971. DOI: 10.1109/TDSC.2015.2502240 (cit. on p. 21).

[80] J. G. Elerath and M Pecht, "Enhanced Reliability Modeling of RAID Storage Systems", in *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, 2007, pp. 175–184. DOI: 10.1109/DSN.2007.41 (cit. on p. 21).

[81] M. Grottke and K. Trivedi, "Fighting bugs: remove, retry, replicate, and rejuvenate", *Journal of Computer*, vol. 40, no. 2, pp. 107–109, 2007, ISSN: 0018-9162. DOI: 10.1109/MC.2007.55 (cit. on p. 22).

[82] K. S. Trivedi, "Availability Analysis of Cisco GSR 12000 and Juniper M20/M40", Tech. Rep., 2000 (cit. on p. 23).

[83] Cisco, "Implementing Access Lists on Cisco 12000 Series Internet Routers", Tech. Rep., 2007 (cit. on p. 23).

[84] ——, "Cisco 12000 Series Internet Router Architecture: Route Processor", Tech. Rep., 2005 (cit. on p. 23).

[85] ——, "Cisco 12000 Series Internet Router Architecture: Line Card Design", Tech. Rep., 2005 (cit. on p. 23).

[86] ——, "Cisco 12000 Series Internet Router Architecture: Line Card Design", Tech. Rep., 2005 (cit. on p. 23).

[87] ——, "Cisco 12000 Series Internet Router Architecture: Route Processor", Tech. Rep., 2005 (cit. on p. 23).

[88] ——, "Cisco 12000 Series Internet Router Architecture: Chassis", Tech. Rep., 2006 (cit. on p. 24).

[89] M. Melo, P. Maciel, J. Araujo, R. Matos, and C. Araujo, "Availability study on cloud computing environments: Live migration as a rejuvenation mechanism", in *2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, IEEE, 2013, pp. 1–6, ISBN: 978-1-4673-6472-0. DOI: 10.1109/DSN.2013.6575322 (cit. on p. 25).

[90] H. S. Kishor S. Trivedi, K. S. Trivedi, and H. Sun, "Stochastic Petri Nets and Their Applications to Performance Analysis of Computer Networks", pp. 1–27, 1998 (cit. on p. 25).

[91] M. O. Ball, "Computing Network Reliability", *Operations Research*, vol. 27, no. 4, pp. 823–838, 1979, ISSN: 0030-364X. DOI: 10.1287/opre.27.4.823 (cit. on p. 25).

[92] M. O. Ball, C. J. Colbourn, and J. S. Provan, "Network reliability", in, 1995, pp. 673–762. DOI: 10.1016/S0927-0507(05)80128-8 (cit. on p. 25).

[93] A. Agrawal and R. E. Barlow, "A Survey of Network Reliability and Domination Theory", *Operations Research*, vol. 32, no. 3, pp. 478–492, 1984, ISSN: 0030-364X. DOI: 10.1287/opre.32.3.478 (cit. on p. 25).

[94] W. Li, I. Santos, F. C. Delicato, P. F. Pires, L. Pirmez, W. Wei, H. Song, A. Zomaya, and S. Khan, "System modelling and performance evaluation of a three-tier Cloud of Things", *Future Generation Computer Systems*, vol. 70, pp. 104–125, 2017, ISSN: 0167739X. DOI: 10.1016/j.future.2016.06.019 (cit. on p. 37).

[95] H. Choi and K. Trivedi, "Approximate performance models of polling systems using stochastic Petri nets", in *[Proceedings] IEEE INFOCOM '92: The Conference on Computer Communications*, IEEE, 1992, 2306–2314 vol.3, ISBN: 0-7803-0602-3. DOI: 10.1109/INFCOM.1992.263520 (cit. on p. 37).

[96] Y. Ma, J. Han, and K. Trivedi, "Composite performance and availability analysis of communications networks. A comparison of exact and approximate approaches", English, in *Globecom '00 - IEEE. Global Telecommunications Conference. Conference Record (Cat. No.00CH37137)*, vol. 3, IEEE, 2000, pp. 1771–1777, ISBN: 0-7803-6451-1. DOI: 10.1109/GLOCOM.2000.891940 (cit. on p. 37).

[97] R. Ghosh, F. Longo, V. K. Naik, and K. S. Trivedi, *Modeling and performance analysis of large scale IaaS Clouds*, 2012. DOI: 10.1016/j.future.2012.06.005 (cit. on p. 37).

[98] M. Torquato and M. Vieira, "Interacting SRN Models for Availability Evaluation of VM Migration as Rejuvenation on a System under Varying Workload", in *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, IEEE, 2018, pp. 300–307, ISBN: 978-1-5386-9443-5. DOI: 10.1109/ISSREW.2018.00022 (cit. on p. 37).

[99] D. S. Kim, J. B. Hong, T. A. Nguyen, F. Machida, K. S. Trivedi, and J. Park, "Availability Modeling and Analysis of a Virtualized System using Stochastic Reward Nets", in *Proceedings of 16th IEEE International Conference on Computer and Information Technology (IEEE CIT 2016)*, Fiji: IEEE, 2016, ISBN: 9781509043149. DOI: 10.1109/CIT.2016.97 (cit. on p. 37).

[100] K. Trivedi, E. Andrade, and F. Machida, "Combining Performance and Availability Analysis in Practice", in *Advances in Computers*, 1st ed., vol. 84, Amsterdam, The Netherlands: Academic Press, 2012, ch. 1, pp. 1–38, ISBN: 9780123965257. DOI: 10.1016/B978-0-12-396525-7.00001-0 (cit. on p. 38).

[101] R. Xia, X. Yin, J. Alonso Lopez, F. Machida, and K. S. Trivedi, "Performance and availability modeling of IT systems with data backup and restore", *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 4, pp. 375–389, 2014, ISSN: 15455971. DOI: 10.1109/TDSC.2013.50 (cit. on p. 38).

[102] G. P. Katerina and K. S. Trivedi, "Stochastic Modeling Formalisms for Dependability, Performance and

Performability", *Performance Evaluation: Origins and Directions*, pp. 403–422, 2000 (cit. on p. 38).

[103] P. Buchholz, "An adaptive aggregation/disaggregation algorithm for hierarchical Markovian models", *European Journal of Operational Research*, vol. 116, no. 3, pp. 545–564, 1999, ISSN: 03772217. DOI: 10.1016/S0377-2217(98)00088-5 (cit. on p. 38).

[104] M. Ajmone Marsan, R. Gaeta, and M. Meo, "Accurate approximate analysis of cell-based switch architectures", *Performance Evaluation*, vol. 45, no. 1, pp. 33–56, 2001, ISSN: 01665316. DOI: 10.1016/S0166-5316(00)00064-X (cit. on p. 38).

[105] I. Davies, W. J. Knottenbelt, and P. S. Kritzinger, "Symbolic methods for the state space exploration of GSPN models", in *Proceedings of 12th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation (TOOLS'02)*, 2002, pp. 188–199 (cit. on p. 38).

[106] A. Miner, "Efficient state space generation of GSPNs using decision diagrams", in *Proceedings International Conference on Dependable Systems and Networks*, IEEE Comput. Soc, 2002, pp. 637–646, ISBN: 0-7695-1597-5. DOI: 10.1109/DSN.2002.1029009 (cit. on p. 38).

[107] P. Ballarini, S. Donatelli, and G. Franceschinis, "Parametric stochastic well-formed nets and compositional modelling", in *Proceedings of the 21st international conference on Application and theory of Petri Nets (ICATPN'00)*, Springer-Verlag, 2000, pp. 43–62, ISBN: 3-540-67693-7 (cit. on p. 38).

[108] 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, ISSN: 01665316. DOI: 10.1016/S0166-5316(99)00061-9 (cit. on p. 38).

• • •