Availability Analysis of IP Multimedia Subsystem in Cloud Environments

Mario Di Mauro, Giovanni Galatro, Maurizio Longo, Fabio Postiglione
Dept. of Information and Elect. Eng. and Applied Maths (DIEM)
University of Salerno (Italy)
{mdimauro,longo,fpostiglione}@unisa.it, galatrogiovanni@gmail.com

Marco Tambasco
Research Consortium on Telecommunications
CoRiTeL, Fisciano (SA), Italy
marco.tambasco@coritel.it

***Abstract*—As of today, telecommunication providers are exploiting the possibilities offered by the cloud paradigm to efficiently decouple physical network resources, like hardware equipment, optical interfaces and cables, from offered services, for instance multimedia content delivery and data storage. Among technologies conceived to implement this paradigm, containerization stands out. It can be considered as an evolution of classic virtualization, where software instances called *containers* are designed to offer specific network functionalities by relying on a separate infrastructure composed of virtual machines and hardware. In line with this new trend, we characterize, from an availability viewpoint, an IP Multimedia Subsystem (IMS) architecture deployed in a containerized environment (dubbed cIMS), which represents a pivotal part of novel network architectures such as 5G. Firstly, we model the availability of cIMS by employing both Reliability Block Diagram (RBD), to capture logical dependencies among cIMS nodes, and Stochastic Reward Networks (SRN), to characterize individually the probabilistic behavior of each node. Then, also supported by an ad-hoc automated procedure, we carry out an experimental assessment of a typical telecommunication network service satisfying a desired availability constraint, whose results are some feasible cIMS configurations that can be deployed.**

***Index Terms*—IP Multimedia Subsystem, Containers, Availability Analysis, Stochastic Reward Nets, Redundancy Optimization.**

I. Introduction and Related Work

Cloud concepts are conquering the telecommunication world since they promise appealing advantages such as flexibility and adaptability in deploying network infrastructures, ease in systems management, significant cost saving. Techniques such as virtualization and containerization provide the grounds to deploy cloud-based networks. The latter, in particular, is based on the concept of *container*, a lightweight software instance [1] that can embed a specific network functionality to provide (e.g. routing, firewalling, switching, etc.). In this way, service layer (implemented via containers) and physical layer (realized through virtual machines and shared hardware) are decoupled so to guarantee a quick time-to-market. Among network architectures which benefit from this approach there is IP Multimedia Subsystem (IMS), designed to handle multimedia content across new generation networks. Starting from availability concepts applied also in other contexts ([2], [3], [4]), we consider a containerized version of IMS (called cIMS) which we characterize in terms of availability. Precisely, we present a two-layer hierarchical model where: the upper layer, descriptive of logical interconnections among cIMS subsystems, benefits from Reliability Block Diagram (RBD) representation, whereas the lower layer, capturing the stochastic behavior of single cIMS subsystems that undergo failure and repair events, benefits from Stochastic Reward Networks (SRN) methodology.

We describe an experimental evaluation (with the support of a designed-from-scratch routine and of a cIMS testbed deployed in lab) designed to pinpoint a set of cIMS configurations satisfying specific availability criteria such as "four nines" (maximum tolerated cIMS downtime of 26 minutes and 30 seconds per year) or "five nines" (maximum tolerated cIMS downtime of 5 minutes and 26 seconds per year).

In the last years, an increasing interest on the availability and reliability issues of novel network infrastructures is noticed. In [5] an availability assessment of cloud-based LTE (Long Term Evolution) nodes is proposed, by relying on Stochastic Activity Networks formalism. In [6], Stochastic Petri Networks are exploited to characterize redundancy in virtualized settings; the same formalism is adopted in [7] to characterize availability of a cloud environment which realizes a Disaster-Recovery-as-a-Service framework. The problem of minimum virtual elements to deploy in order to avoid failures in virtualized environments is faced in [8], while [9] and [10] identify the optimal distribution of virtualized functions to minimize failure events. Other works are aimed at designing frameworks to deal with the availability issues: it is the case of [11], proposing a restoration method for Openstack, a cloud-based operating system, or [12], presenting a platform for fault management of virtualized environments.

In our work, branching from some recent contributes such as [13], [14], [15], we offer an availability characterization of a cIMS, considered the state-of-the-art in terms of network softwarization.

The paper adheres to the following organization. In Section II we detail the functionalities of the IMS architecture deployed in a containerized environment; Section III introduces the availability model of cIMS along with details about RBD and SRN formalisms; in Section IV, we propose an availability assessment by comparing some cIMS configurations (or settings) which differ in terms of availability outcomes and deployment costs. Finally, Section V summarizes the results and envisages some hints for future analyses.

II. IMS IN CLOUD: THE CASE OF CONTAINERIZED ENVIRONMENT

IP Multimedia Subsystem [16] is undoubtedly a crucial part in the scenario of novel network deployments, including 5G. It has been introduced to handle multimedia sessions (e.g. HD Voice/Video) by means of SIP protocol and some specific nodes such as *i)* Proxy CSCF (P-CSCF) providing SIP proxy functionalities, *ii)* Serving CSCF (S-CSCF) devoted at handling crucial functions such as session management and control, *iii)* Interrogating CSCF (I-CSCF) responsible to forward SIP requests/responses across the IMS domain, *iv)* Home Subscriber Server (HSS) in charge of managing users profiles by means of a set of databases (possibly distributed).

IMS well fits to be deployed in cloud environments involving virtualization or containerization paradigms, granting the possibility to "softwarize" network nodes. Precisely, the latter paradigm offers a winning solution with respect to the one provided by virtual machines, due to the more flexible and cost-saving deployment without sacrificing performance.

It is useful to remark some differences between virtualization and containerization paradigms. Actually, the final purpose of both is the softwarization of services, namely, the abstraction at a software level of network functionalities provided by hardware nodes (e.g. routers, switches, firewalls, etc). Such an abstraction is realized through an interface between hardware equipment and software layer called hypervisor. In case of virtualization, hypervisor hosts virtual machines each one equipped with its own operating system. In case of containerization (a.k.a. lightweight virtualization), software entities called containers do not need an operating system since they are managed by a container manager (Docker is the most popular and used example), thus, the softwarization process is more comfortable and easy. By contrast, if not designed well, container architectures might suffer from the lack of a fully isolated environment which makes them less secure than virtualized infrastructures.

We consider a containerized deployment of an IMS framework (cIMS) where, for each node, we distinguish two layers: the Containerized Network Function (CNF) layer which embodies a logical abstraction of a specific IMS functionality (e.g. I-CSCF, HSS, etc.); the Containerized Network Replica (CNR) layer which represents a physical deployment of a CNF. Figure 1 clarifies the mapping between these two layers; it also shows that CNFs can be implemented by means of more than one CNR.

The forthcoming availability characterization assumes a five-level structure of a CNR including (from the bottom to the top):

- an infrastructure level (HW) which comprises hardware supplies such as CPU, RAM, etc.
- a hypervisor (HPV) level providing a separation interface between hardware (upper) and software-based (lower) levels;
- a virtual machine (VM) level hosting the containerized environment (Docker in our case);

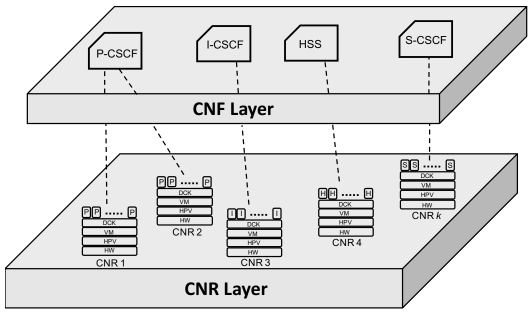


Fig. 1: Two-layer model: CNF (logical abstraction) and CRN (physical deployment).

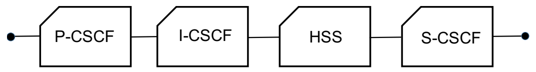


Fig. 2: Reliability Block Diagram characterizing logical interconnections among (containerized) IMS nodes.

- a Docker level (DCK) which provides a runtime environment to manage containers;
- a Container (CNT) level concentrating the specific software functionality (Proxy, Serving, HSS).

According to such representation, redundancy can be obtained in two ways: at CNT level, by increasing the number of containers (up to a load threshold) on top of a CNR, and at CNR level by replicating the whole CNR structure.

III. AVAILABILITY CHARACTERIZATION

Our aim is to determine the steady-state availability A of the cIMS, namely the fraction of time wherein the system is fully working. By virtue of ergodicity, it can be expressed as

$$A = \lim_{t \to +\infty} A(t), \tag{1}$$

where $A(t)$ is the so called instantaneous availability, namely the probability that system is working at time t. In turn, $A(t)$ is usually expressed in terms of another random process $Z(t)$, called *reward function*, having the following meaning: $Z(t) = 1$ if the system is working ("up" condition) at time t, and $Z(t) = 0$ if it is not working ("down" condition) at time t, so that:

$$A(t) = \Pr\{Z(t) = 1\} = E[Z(t)]. \tag{2}$$

Our availability analysis relies on a hierarchical decomposition based on two complementary system models: *i)* RBD to model the IMS service logic by cIMS nodes that interwork; *ii)* SRN to characterize single CNRs availability, by modeling all its layers and their interactions.

The SRN model representative of a single CNR is depicted in Fig. 3, where the following graphical elements are present:

- *Places*: drawn by circles, denote a system working condition (e.g. a layer up or down), where, inside each place, one or more *tokens* (indicated by numbers or symbols) denote an holding condition. In Fig. 3 one can recognize the following places: P_{upCNT} $[P_{dnCNT}]$, P_{upDCK}

$[P_{dnDCK}]$, P_{upVM} $[P_{dnVM}]$, P_{upHPV} $[P_{dnHPV}]$, and P_{upHW} $[P_{dnHW}]$ which accounts for the working [failure] conditions of container, docker, virtual machine, hypervisor, and hardware, respectively.

- *Timed Transitions*: drawn by unfilled and thick rectangles, are representative of actions that might occur, such as failures and consequent repairs, and are characterized by exponentially distributed times[1]. In Fig. 3 it is also possible to pinpoint the following transitions: T_{fCNT} $[T_{rCNT}]$, T_{fDCK} $[T_{rDCK}]$, T_{fVM} $[T_{rVM}]$, T_{fHPV} $[T_{rHPV}]$, and T_{fHW} $[T_{rHW}]$ which take into account failure [repair] actions involving containers, docker, virtual machine, hypervisor, and hardware, respectively.
- *Immediate Transitions*: drawn by filled and thin rectangles, denote actions occurring in a (nearly) zero-length time interval. In Fig. 3, the following immediate transitions are recognizable: t_{CNT}, t_{DCK}, t_{VM}, and t_{HPV}, which account for instantaneous events pertaining to container, docker, virtual machine, hypervisor levels, respectively.

When analyzing the dynamics of an SRN, a failure/repair event results in *firing* a timed transition with the consequence that a token is moved from its source place to the destination place. It is worth noting that, in our case, the only place containing more than one token is the CNT level, due to the fact that more containers can be hosted on a single CNR.

Letting S be the set of *markings*, namely distributions of tokens in the given SRN, $E[Z(t)]$, and hence $A(t)$, can be expressed as

$$E[Z(t)] = \sum_{i \in S} r_i(t) \cdot p_i(t), \tag{3}$$

where, $r_i(t)$ is the *reward rate* associated to marking i, that is equal to 1 if marking corresponds to an up condition at time t and 0 otherwise, whereas $p_i(t)$ is the occurrence probability of marking i at time t (see [17] for further discussion). Finally,

$$A = \lim_{t \to +\infty} E[Z(t)] = \sum_{i \in S} r_i \cdot p_i, \tag{4}$$

where $r_i = \lim_{t \to +\infty} r_i(t)$ is the steady-state reward rate, and $p_i = \lim_{t \to +\infty} p_i(t)$ is the steady-state probability of marking i.

Let us now virtually follow the evolution of SRN depicted in Fig. 3. Supposing a single container failure (caused, for example, by an unpredictable container reboot) transition T_{fCNT} is fired and one token in P_{upCNT} moves to P_{dnCNT}. As a consequence, $n_k - 1$ tokens lie in P_{upCNT}, where n_k is the initial number of tokes in P_{upCNT}. By contrast, once the container is rebooted, T_{rCNT} is fired and the token comes back to place P_{upCNT}, and the counter of tokens is increased by 1. If we consider a docker level failure, once T_{fDCK} is fired, the (only one) available token is transferred from P_{upDCK} to P_{dnDCK}. Now, due to the DCK level failure, no

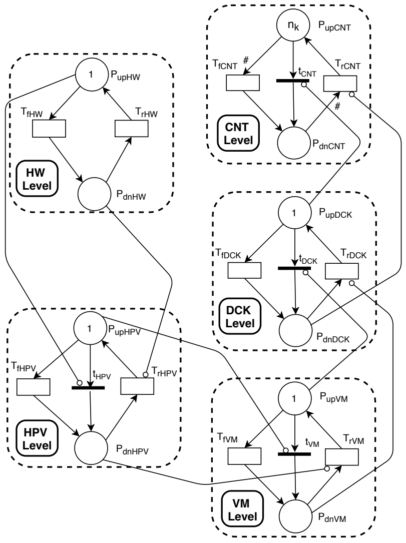


Fig. 3: CNR modeled in terms of an SRN.

container would be active, thus, an inhibitory arc (depicted as a little segment between P_{upDCK} and t_{CNT} with a little circle at the far end) drives t_{CNT} to be fired. As docker level gets repaired, T_{rDCK} is fired, thus: *i)* the token comes back from P_{dnDCK} to P_{upDCK}, and *ii)* the inhibitory arc between P_{dnDCK} and T_{rCNT} is deactivated, so that n_k tokens can be moved again from P_{dnCNT} to P_{upCNT}. Similar considerations hold when describing failure/repair events occurring at the remaining levels.

It is useful to introduce:

- *demand* W: required system performance, in terms of simultaneous IMS sessions to manage;
- *capacity* c_j: maximum number of simultaneous IMS sessions to be handled by a single container on CNF j.

A generic node is working at time t when the number of manageable IMS sessions $c_j \cdot \sum_{h=1}^{k} \#P^{(h)}_{upCNT}(t)$ is greater than demand W, where h represents the number of CNRs (ranging from 1 CNR per node to k CNRs per node), whereas "#" denotes the number of tokens.

By denoting the normalized performance level as $\gamma_j = W/c_j$, one can express the reward rate of node j in marking i at time t as

$$r_{ij}(t) = \begin{cases} 1 & \text{if} \quad \sum_{h=1}^{k} \#P^{(h)}_{upCNT}(t) \geq \gamma_j, \\ \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Through an obvious adaptation of (4), the *steady-state availability* of CNF j can be calculated as:

$$A_j = \sum_{i \in S} r_{ij} \cdot p_{ij}. \tag{6}$$

Finally, it is possible to derive the steady-state availability of the whole cIMS system, viz.

$$A_{cIMS} = \prod_{j=1}^{L} A_j, \tag{7}$$

[1]This represents a well accepted assumption when dealing with availability assessments of network deployments - see [7], [23], [24].

where L is number of subsytems to be connected to offer the given cIMS service. The product in (7) is a consequence of the series structure of cIMS system described by the RBD scheme in Fig. 2 (with $L = 4$), since the node chain works when each node works.

IV. Availability Assessment

The following assessment is aimed at identifying the cIMS configurations satisfying required availability constraints, and benefits from the interaction with TimeNET [18], a tool designed to build SRN models that we automatically recall by means of a designed-from-scratch routine written in Python.

Before going forward, it is useful to clarify the concept of CNR "cost" that we use across such analysis. As often occurs for *layered* services offered by top-players such as Amazon AWS and Microsoft Azure, we assign to a CNR a cost composed of the following (dimensionless and customizable) contributions all amounting to 1: *i)* cost per container; *ii)* cost per docker/VM; *iii)* cost per hardware/hypervisor. Our approach can be immediately extended to cope with different costs, according to specific needs.

A. The experimental Testbed

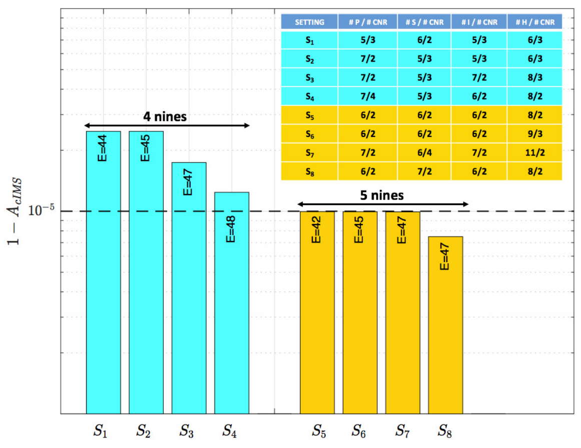
In our analysis, we refer to a fixed performance demand $W = 3000$ and $c = 1000$ by assuming $c_j = c$. Such a hypothesis is supported by load tests carried out through Clearwater platform [19], a containerized deployment of IMS. More precisely, on a laptop based on an Intel Core CPU i7 − 3630QM@2.40GHz and with a RAM of 8 GB, we deploy two virtual machines (1 virtual Core and 2 GB of RAM per VM): the first one serves as a containerized deployment of the whole cIMS architecture including P-CSCF (*Bono*), S/I-CSCF (*Sprout*), and HSS (*Homestead*). The second VM is a stress node that embeds some routines useful to perform a load stress against the containerized platform. The test scenario envisages 1000 IMS sessions with a BHCA (Busy Hour Call Attempts) equal to 2.6 per user (in line with values provided for VoLTE - see [20]), and where the resulting average call setup delay is 80 msec. Such a value is reasonable by considering that the architecture is deployed on the same node, thus, interconnection delays are negligible.

B. Numerical Analysis

The presented numerical analysis relies on the assumption that CSCF nodes are equally treated. The only exception is the HSS node that, hosting a database of customers, is considered the most crucial element. To take into account such a level of criticality, we admit slightly different values for γ_{CSCF} and γ_{HSS}, namely $\gamma_{CSCF} = \gamma = W/c = 3$, and $\gamma_{HSS} = \gamma + 1 = 4$. In short, it means that we are considering an *extra* CNR for the HSS representation, recalling classic hot/standby database configurations. As regards mean-time-to-failure (MTTF) and mean-time-to-repair (MTTR) parameters of some container-based structures, we refer to the only (at now) available dataset released by Google already processed by [21]. For remaining

TABLE I: Parameters values.

Parameter	Description	Value
$1/\lambda_{CNT}$	mean time for container failure	500 hours
$1/\lambda_{DCK}$	mean time for docker daemon failure	1000 hours
$1/\lambda_{VM}$	mean time for virtual machine failure	2880 hours
$1/\lambda_{HPV}$	mean time for hypervisor failure	2880 hours
$1/\lambda_{HW}$	mean time for hardware failure	60000 hours
$1/\mu_{CNT}$	mean time for container repair	2 secs
$1/\mu_{DCK}$	mean time for docker daemon repair	5 secs
$1/\mu_{VM}$	mean time for virtual machine repair	1 hours
$1/\mu_{HPV}$	mean time for hypervisor repair	2 hours
$1/\mu_{HW}$	mean time for hardware repair	8 hours
W	performance demand	3000
c	performance capacity	1000 IMS sessions
A_0	steady-state availability requirement	0.9999 / 0.99999



SETTING	# P / # CNR	# S / # CNR	# I / # CNR	# H / # CNR
S_1	5/3	6/2	5/3	6/3
S_2	7/2	5/3	5/3	6/3
S_3	7/2	5/3	7/2	8/3
S_4	7/4	5/3	6/2	8/2
S_5	6/2	6/2	6/2	8/2
S_6	6/2	6/2	6/2	9/3
S_7	7/2	6/4	7/2	11/2
S_8	6/2	7/2	6/2	8/2

Fig. 4: Steady-state unavailability analysis considering (S_1, …,S_4) settings satisfying "four nines" requirement, and (S_5, …,S_8) settings satisfying "five nines".

parameters, we refer to classic technical literature (see e.g. [22]) with values reported in Table I.

The proposed analysis exploits a procedure allowing to: *i)* automatically build SRN/RBD models for the availability characterization of cIMS, *ii)* extract a selection of settings (namely, feasible cIMS configurations) satisfying availability conditions (e.g. $A_{cIMS} = 0.9999$ ("four nines")) or $A_{cIMS} = 0.99999$ ("five nines")) along with indication of costs E (expenditure).

In Fig. 4 we report the unavailability ($1 - A_{cIMS}$) results for 8 exemplary settings. Precisely, settings (S_1, …,S_4), in light blue, are chosen among ones satisfying the "four nines" requirement, whereas settings (S_5, …,S_8), in orange, are selected among ones satisfying "five nines". The horizontal dashed line at 10^{-5} delimits the separation between the two classes of settings. Inside the bar graph we include a table, where, the first column indicates the number of setting, and the remaining columns indicate how many containers/CNRs are deployed. For instance, in case of S_1 the second column indicates that 5 P-CSCF containers are deployed across 3 CNRs (5/3). Inside each bar we also report the cost (E) associated with each setting. According to the costs assignment criterion explained at the beginning of Sect. IV, the total cost for setting e.g. S_1 can be built as follows:

- E_P (cost for P-CSCF) $= 1 \cdot 5$ (CNT) $+1 \cdot 3$ (DCK+VM)

$+1 \cdot 3$ (HPV+HW)= 11;

- E_S (cost for S-CSCF) $= 1 \cdot 6$ (CNT) $+1 \cdot 2$ (DCK+VM) $+1 \cdot 2$ (HPV+HW)= 10;
- E_I (cost for I-CSCF) $= 1 \cdot 5$ (CNT) $+1 \cdot 3$ (DCK+VM) $+1 \cdot 3$ (HPV+HW)= 11;
- E_H (cost for HSS) $= 1 \cdot 6$ (CNT) $+1 \cdot 3$ (DCK+VM) $+1 \cdot 3$ (HPV+HW)= 12.

Consequently, the final cost for setting S_1 amounts to: $E = E_P + E_S + E_I + E_H = 44$.

Some interesting considerations can be made about the obtained results. Usually, higher costs reflect more redundant settings that, in turn, correspond to configurations of higher availability. This general rule can be violated by choosing different distributions of containers and CNRs. In fact, let us compare S_4 and S_5. The former allows to satisfy the "four nines" at a cost of 48, whereas, the latter guarantees "five nines" condition at the lower cost of (42). This is due to an unfavourable container/CNR distribution for S_4, since more CNRs are used for P-CSCF and for S-CSCF nodes w.r.t. the same nodes in S_5. In fact, as a CNR embodies two cost contributions ((DCK+VM) and (HPV+HW)), this explains the cost differences between S_4 and S_5. For the same reason, some settings can exhibit different availability values being costs the same. It is the case of S_7 and S_8 settings, where $A_{cIMS} = 0.99999007$ and $A_{cIMS} = 0.99999251$, respectively. Obviously, a designer would choose the latter setting since it is more robust to failures (higher availability).

V. Conclusions

Among cloud-based enabling technologies, containers represent the state-of-the-art, since, unlike usual virtual machines, they do not need an operating system to work, being directly hosted on virtualized environments. In particular, containers can embody network functionalities which a telecom provider can tune and manage in a very ductile way. A framework such as the IP Multimedia Subsystem (IMS), born to handle advanced services (e.g. HD Voice/Video) across 5G arrangements, is well suited to be deployed in containerized environments (cIMS). We have provided an availability assessment of a cIMS that passes through two steps. The first one is aimed at modeling a cIMS infrastructure by considering two techniques: Reliability Block Diagram (RBD) and Stochastic Reward Nets (SRN), helpful to capture macroscopic interconnections among nodes and stochastic behaviors, respectively. The second step has consisted in carrying out an experimental evaluation with the aim of selecting a subset of cIMS configurations (or settings) satisfying availability criteria such as "four nines" or "five nines". This latter step has exploited a Python-based procedure, realized to automate the process of building and selecting feasible settings, and a realistic testbed helpful to validate some assumptions on cIMS load. In the future, such work can be extended to characterize more sophisticated scenarios, coping, for instance, with the presence of time-varying load constraints, or with multi-tenant deployments in which several providers share a single cIMS node, as it occurs in modern slice-based networks.

References

[1] Y. Zhang. *Network Function Virtualization:Concepts and Applicability in 5G Networks.* Wiley-IEEE Press, Inc., 1st ed., 2018.

[2] M.A. Mellal, E. Zio. Availability Optimization of Parallel-Series System by Evolutionary Computation. In *2018 IEEE International Conference on System Reliability and Safety (ICSRS)*, pages 198–202, 2018.

[3] S. Du, E. Zio, R. Kang, "A New Analytical Approach for Interval Availability Analysis of Markov Repairable Systems," *IEEE Trans. Rel.*, vol. 67, no. 1, pp. 118–128, 2017.

[4] W. Wang, F. Di Maio, E. Zio, "Adversarial Risk Analysis to Allocate Optimal Defense Resources for Protecting Cyber-Physical Systems from Cyber Attacks," *Risk analysis : an official publication of the Society for Risk Analysis*, DOI:10.1111/risa.13382, 2019.

[5] A. Gonzalez, P. Gronsund, K. Mahmood, B. Helvik, P. Heegaard, and G. Nencioni. Service availability in the NFV virtualized evolved packet core. In *2015 IEEE GLOBECOM*, pages 1–6, 2015.

[6] S. Fernandes, E. Tavares, M. Santos, V. Lira, and P. Maciel. Dependability assessment of virtualized networks. In *Proc. IEEE ICC 2012*, pages 2711–2716, 2012.

[7] E. Andrade, B. Nogueira, R. Matos, G. Callou, and P. Maciel, "Availability modeling and analysis of a disaster-recovery-as-a-service solution," *Computing*, vol. 99, no. 10, pp. 929–954, 2017.

[8] J. Liu, Z. Jiang, N. Kato, O. Akashi, and A. Takahara, "Reliability evaluation for NFV deployment of future mobile broadband networks," *IEEE Wireless Commun.*, vol. 23, no. 3, pp. 90–96, 2016.

[9] J. Fan, C. Guan, Y. Zhao, and C. Qiao, "Availability-aware mapping of service function chains," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pp. 1–9, 2017.

[10] J. Kong, I. Kim, X. Wang, Q. Zhang, H. C. Cankaya, W. Xie, T. Ikeuchi, and J. P. Jue, "Guaranteed-availability Network Function Virtualization with Network Protection and VNF replication," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pp. 1–6, 2017.

[11] Y. Yamato, Y. Nishizawa, S. Nagao, and K. Sato. Fast and reliable restoration method of virtual resources on OpenStack. *IEEE Transactions on Cloud Computing (In Press)*, 6(2):572–583, 2015.

[12] H. B. Lee, S. I. Kim, and H. S. Kim. A fault management system for NFV. In *2018 International Conference on Information Networking (ICOIN)*, pages 640–645, 2018.

[13] M. Di Mauro, M. Longo, and F. Postiglione, "Availability Evaluation of Multi-tenant Service Function Chaining Infrastructures by Multidimensional Universal Generating Function," *IEEE Trans. Serv. Comput.*, DOI: 10.1109/TSC.2018.2885748, 2018.

[14] M. Di Mauro, G. Galatro, M. Longo, F. Postiglione, and M. Tambasco, "Availability evaluation of a virtualized IP multimedia subsystem for 5G network architectures", in *ESREL 2017*, (Portorose, Slovenia, pp. 2203–2210, Jun. 2017).

[15] M. Di Mauro, G. Galatro, M. Longo, F. Postiglione, and M. Tambasco, "Availability modeling of a virtualized IP multimedia subsystem using non-Markovian stochastic reward nets", in *ESREL 2018*, (Trondheim, Norway, pp. 2427–2435, Jun. 2018).

[16] G. Camarillo, and M.A. Garcia-Martin *The 3G IP Multimedia Subsystem.* New York, John Wiley and Sons, Inc., 3rd ed., 2008.

[17] J.K. Muppala, G. Ciardo, and K.S. Trivedi, "Stochastic Reward Nets for Reliability Prediction," in *Communications in Reliability, Maintainability and Serviceability*, pp. 9–20, 1994.

[18] R. German, C. Kelling, A. Zimmermann, and G. Hommel, "TimeNET: a toolkit for evaluating non-Markovian stochastic Petri nets," *Performance Evaluation*, vol. 24, no. 1-2, pp. 69–87, 1995.

[19] The Clearwater Project. Available: http://www.projectclearwater.org.

[20] Tonse Telecom, "The LTE Data Storm in the Core of Your Network", White Paper, Jan. 2013.

[21] S. Sebastio, R. Ghosh, and T. Mukherjee, "An availability analysis approach for deployment configurations of containers," *IEEE Trans. Serv. Comput.*, vol. PP, no. 99, pp. 1–1, 2018.

[22] R. Matos, P. Maciel, F. Machida, D. S. Kim, K.S. Trivedi, "Sensitivity analysis of server virtualized system availability," *IEEE Trans. Rel.*, vol. 4, no. 61, pp. 994–1006, 2012.

[23] D. Bruneo, "A Stochastic Model to Investigate Data Center Performance and QoS in IaaS Cloud Computing Systems ," *IEEE Trans. Parallel Distrib. Syst*, vol. 25, no. 3, pp. 560–569, 2014.

[24] G. Nencioni, B. E. Helvik, P. E. Heegaard, "Including Failure Correlation in Availability Modeling of a Software-Defined Backbone Network," *IEEE Trans. Netw. Serv. Man.*, vol. 14, no. 4, pp. 1032–1045, 2017.