

Performability Management of Softwarized IP Multimedia Subsystem

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, g.galatro1@studenti.unisa.it

Marco Tambasco
Research Consortium on Telecommunications
Via Giovanni Paolo II, Fisciano (SA), Italy
marco.tambasco@coritel.it

Abstract—IP Multimedia Subsystem (IMS) represents a crucial element for the convergence of telecommunication systems heading towards 5G solutions. Actually, IMS offers a standardized and vendor independent model allowing network providers to supply multimedia services such as video streaming or HD voice, with pressing QoS requirements. The IMS architecture can dramatically improve its flexibility when deployed within softwarized environments, in conjunction with virtual or container-based technologies. This latter, in particular, realizes an abstraction of software resources from the underlying hardware in a more efficient and resource saving manner than virtual machines. Inspired by this model, we propose a tool for the performability management of IMS architectures deployed in a containerized environment (dubbed cIMS). First, we model the stochastic behavior of a cIMS by means of two complementary formalisms: *i*) Reliability Block Diagram (RBD) amenable to model high level interconnections among cIMS nodes, and *ii*) Stochastic Reward Networks (SRN) useful to capture deeper details of a single node in terms of failure and repair events. Then, we develop an automated procedure aimed at supporting the performability management of cIMS deployments that must satisfy the optimal trade-off among high availability requirements, capacity load, and deployment costs.

Index Terms—Softwarized IP Multimedia Subsystem, Performability, Stochastic Reward Nets, Redundancy Optimization.

I. INTRODUCTION

Network Softwarization refers to those tools, systems, and procedures protagonist of the transformation process that is leading towards 5G network infrastructures [1]. A key enabler of this process is undoubtedly the Network Function Virtualization (NFV) paradigm, in charge of providing advantages in terms of flexibility of resource management and service composition, along with considerable deployment costs reduction [2]. Recently, classic virtualization paradigm is giving way to container-based infrastructures offering a lightweight environment for hosting novel network services [3], such as Service Function Chains (SFC) which allow a smart composition of softwarized network functionalities [4], [5]. Across the telco world, many frameworks exhibit a *chained* infrastructure. It is the case of IP Multimedia Subsystem (IMS) embedded in the 5G core network to provide QoS-demanding services.

In the proposed scenario, we merge container-based technology and service chained structures to derive a containerized version of IP Multimedia Subsystem (dubbed cIMS) that we

characterize in terms of performability, where, performance and availability are considered in a combined fashion [6]. Remarkably, such an issue is of practical interest as argued in [7]. Precisely, our contributions concern: *i*) a “five nines” availability analysis of the whole cIMS chain, where, failures and repairs events are stochastically characterized through Stochastic Reward Networks (SRNs) and Reliability Block Diagram (RBD) formalisms; *ii*) a performability assessment, where, by means of an automated procedure, we solve an optimization problem aimed at the optimal set of cIMS deployments.

The paper has the following structure: Section 2 proposes a list of related works eliciting differences and similarities with the present study; in Section 3 we propose a brief recall of IMS functionalities by focusing on a container-based deployment calling for the introduction of a layered structure of each cIMS node; Section 4 is devoted at describing the availability model of the cIMS where some insights about adopted formalisms are offered; the performability assessment along with an automated procedure to identify optimal cIMS deployments are afforded in Section 5. Finally, some conclusions are drawn in Section 6.

II. RELATED WORK

Recently, academia and industry made a remarkable effort in improving methods and tools for the evaluation of metrics such as availability and performance of innovative telecommunication systems for network management purposes [8], [9], [10], [11]. In particular, paying heed to performance indicators (usually referred to as Service Level Agreements - SLAs) calls for performability analyses which embrace availability and performance evaluations at the same time.

In the proposed bibliographic *excursus*, two main aspects emerge: *i*) when dealing with availability and redundancy optimization in softwarized environments, not always failure and repair actions are modeled; *ii*) performability analyses are often neglected. Conversely, in the present paper, we take into account both of such aspects.

As regards the first issue, availability concerns related to virtualized SFCs are analyzed in [12], where authors calculate the minimum number of backup VNFs to deploy even if the proposed characterization involves only failures but not repair events. Similarly, repair modeling is neglected in [13],

a work proposing a reliability evaluation of NFV deployments in future mobile broadband networks.

As regards the second issue, an availability analysis of microservice-based architectures involving Google Kubernetes is afforded in [14], where performability assessment is mentioned as future work. An availability study of private IaaS cloud infrastructures has been afforded in [15], where performability is declared as out of the scope. Performability assessments lack also in [16], where Stochastic Petri Nets are adopted to model availability of a cloud-based environment.

In the work presented here, evolving from recent works of same authors ([17], [18]), we jointly consider availability and performance aspects by *i*) affording an availability characterization of a containerized IMS (cIMS) architecture, where RBD is adopted to model the chained structure of a cIMS, and, SRN to capture insights of each single cIMS node in terms of failure and repair events; *ii*) proposing a performability assessment supported by an automated procedure aimed at identifying the best cIMS deployments which satisfy optimal trade-offs among several constraints (high availability, capacity loads, deployment costs).

III. CONTAINERIZED IP MULTIMEDIA SUBSYSTEM

IP Multimedia Subsystem has been designed as a framework in charge of providing access to multimedia IP-based services with well-defined quality of service. Actually, IMS has been elected as a key element in 5G-based infrastructures since it allows inter-operations of Voice over LTE paradigm across network operators' boundaries by offering services such as HD voice, video calling, enriched communications. Basically, IMS functionalities are offered by four nodes: Proxy CSCF (P-CSCF) serving as interface between the user device and the IMS domain; Serving CSCF (S-CSCF) performing functions such as routing management; Interrogating CSCF (I-CSCF) which is in charge of forwarding SIP requests/responses across the domain; Home Subscriber Server (HSS) which acts as a users' profiles database.

IP Multimedia Subsystem is well suited to be deployed in software-based environments, since NFV-based contexts allow to elastically scale in/out resources under the control of MANO (MANagement and Orchestration). In this spirit, we consider a containerized model of IMS (dubbed cIMS) where, also inspired by commercial solutions such as Amazon AWS, containers that offer IMS functionalities run on top of virtual machines often under the supervision of Docker, an opensource container engine able to guarantee isolation among layers in order to preserve security [21].

In our cIMS model, we distinguish two layers: the first one includes the Containerized Network Function (CNF) representing a logical abstraction of an IMS functionality to provide (e.g. P-CSCF, HSS, etc.); the second one is given by Containerized Network Replicas (CNRs) representing the practical deployments of a CNF. For redundancy purposes (useful to satisfy availability requirements), a single CNF can be implemented by means of one or more CNRs as clarified by Fig. 1. Actually, a CNR is a five-level structure which

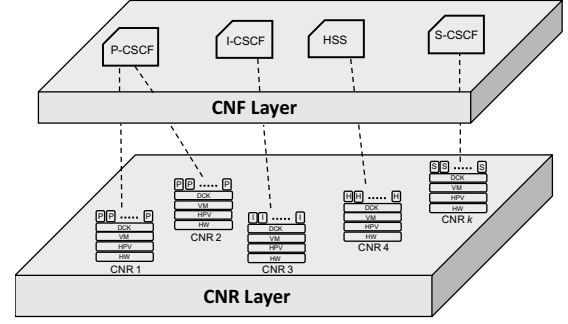


Fig. 1: Logical abstraction of IMS functionalities (CNF Layer) and corresponding deployment (CNR Layer).

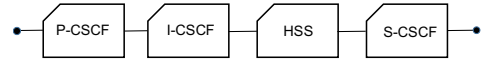


Fig. 2: Reliability Block Diagram modeling the high-level interconnections among nodes of IMS.

includes: *i*) an infrastructure level (HW) which embodies hardware equipment such as power supplies, CPU, RAM; *ii*) a hypervisor (HPV) which provides an interface between hardware and software-based levels; *iii*) a virtual machine (VM) level which hosts Docker environment; *iv*) a Docker daemon (DCK) that provides a runtime ecosystem to handle containers; *v*) Container (CNT) which incorporates the desired software functionality (Proxy, Serving, HSS). It is useful to highlight that, from this representation, the desired redundancy can be obtained at two levels: at CNT level by adding more containers (up to a load threshold) on top of a CNR, and at CNR level by replicating the whole CNR structure.

IV. OVERVIEW OF THE ADOPTED FORMALISMS

The availability model of cIMS passes through the use of RBD and SRN formalisms. The former allows to capture high-level interconnections among nodes over a cIMS chain, as depicted in Fig. 2. The latter [22] is based on Markov Reward Models and is amenable to stochastically characterize the interactions among levels of a CNR (in terms of failure and repair actions) through the description of their state distributions. An SRN can be graphically represented on behalf of a bi-partite directed graph with two basic elements:

- *Place*: depicted as a circle, denotes a specific condition (e.g. up/down) of a level. Inside a place, one or more *tokens* (represented by numbers) account for holding conditions;
- *Transition*: depicted as a rectangle, accounts for actions that might occur (e.g. failures/repairs).

In case a failure/repair event occurs, a transition is *fired* and a token is moved from the source place to the destination place. It is useful to highlight that, in our model, “up” places of various levels can contain one token, except for the CNT level where more tokens are possible, since more containers can be hosted on top of a CNR. Again, transition times are modeled

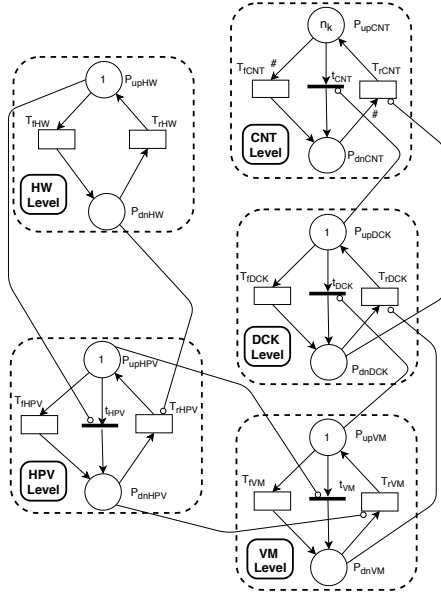


Fig. 3: SRN-based model representative of a generic CNR, along with the indication of the five composing levels.

by means of exponential random variables, a well accepted assumption when dealing with availability assessments [19], [16], [20]. The SRN solution relies on the evaluation of a *reward function*, a non-negative random process which can be mapped with dependability metrics (among them, the availability). Given $Y(t)$ the reward function amounting to 1, when the system is working at time t , and to 0 otherwise, we can express the instantaneous availability as [22]

$$A(t) = Pr\{Y(t) = 1\} = E[Y(t)] = \sum_{i \in S} r_i \cdot p_i(t), \quad (1)$$

where: r_i (commonly referred to as reward rate) represents the value of $Y(t)$ in marking¹ i , $p_i(t)$ indicates the corresponding probability, and S denotes the set of markings. It is possible to split the set S in a subset of “up” states ($r_i = 1$), and a subset of “down” states ($r_i = 0$).

A. An availability model of cIMS

Figure 3 describes the SRN model representative of a CNR. Places P_{upCNT} [P_{dnCNT}], P_{upDCK} [P_{dnDCK}], P_{upVM} [P_{dnVM}], P_{upHPV} [P_{dnHPV}], and P_{upHW} [P_{dnHW}] account for the working [failure] conditions of container instance, docker, virtual machine, hypervisor, and hardware, respectively. Transitions T_{fCNT} [T_{rCNT}], T_{fDCK} [T_{rDCK}], T_{fVM} [T_{rVM}], T_{fHPV} [T_{rHPV}], and T_{fHW} [T_{rHW}] take into account failure [repair] actions involving containers, docker, virtual machine, hypervisor, and hardware, respectively. These transitions (drawn as unfilled rectangles), characterized by exponential distributed times, are called “timed” transitions. By contrast, transitions t_{CNT} , t_{DCK} , t_{VM} , and t_{HPV} (drawn

by filled and thin rectangles), account for actions occurring in a zero-length time interval, and, are called “immediate” transitions. A timed transition with “#” symbol nearby is called “marking-dependent”, meaning that its rate is multiplied by the number of tokens present in the corresponding place. Let us now analyze the temporal evolution of the considered SRN by starting from an initial working condition which envisages: *i*) n_k tokens located in place P_{upCNT} , *ii*) 1 token in all remaining “up” places. In case of a single container failure (e.g. an unexpected container instance reboot), T_{fCNT} is fired and one token in P_{upCNT} is transferred to place P_{dnCNT} . Consequently, $n_k - 1$ tokens remain in P_{upCNT} . On the other hand, once the container is reactivated, T_{rCNT} is fired and the token comes back to P_{upCNT} . In case of docker level failure, instead, T_{fDCK} is fired and the token passes from P_{upDCK} to P_{dnDCK} . Since all container instances fall down until docker level gets repaired, an inhibitory arc (drawn as a segment between P_{upDCK} and t_{CNT} with a little circle closer to the latter) forces t_{CNT} to be fired in case of docker failure. In case of docker repair, transition T_{rDCK} is fired, and two actions are put in place: *i*), the token is moved from P_{dnDCK} to P_{upDCK} , *ii*) the inhibitory arc between P_{dnDCK} and T_{rCNT} is disabled, thus, the n_k tokens are ready to be transferred again from P_{dnCNT} to P_{upCNT} . We omit the description of failures/repairs occurring at VM, HPV, HW levels which can be easily derived from the SRN representation of Fig. 3. We just remark that HW does not admit any immediate transition, representing the lower level of CNR.

It is helpful to introduce two quantities in the forthcoming performability analysis: the *demand* W defined as the required system performance expressed in terms of concurrent IMS sessions; the *capacity* c_j defined as the maximum number of concurrent IMS sessions manageable by a single container deployed on to CNF j . Consequently, the reward rate in marking i can be expressed as

$$r_i(j) = \begin{cases} 1 & \text{if } \sum_{h=1}^k \#P_{upCNT}^{(h)} \cdot c_j \geq W, \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

where “#” indicates the number of tokens². Given $\gamma_j = W/c_j$ the normalized performance level, (2) becomes

$$r_i(j) = \begin{cases} 1 & \text{if } \sum_{h=1}^k \#P_{upCNT}^{(h)} \geq \gamma_j, \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

Then, we can derive the *steady-state availability* for cIMS node j as follows:

$$A_j = \lim_{t \rightarrow +\infty} A_j(t) = \sum_{i \in S} r_i(j) \cdot p_{ij}, \quad (4)$$

where: $r_i(j)$ stems from (3), and p_{ij} represents the steady-state probability expressed by $p_{ij} = \lim_{t \rightarrow +\infty} p_{ij}(t)$ (being $p_{ij}(t)$ the instantaneous probability of node j in marking i).

¹A *marking* represents a possible distribution of tokens.

²In Petri Net jargon, with a little abuse, the symbol # denotes both the number of tokens and the marking-dependent transitions.

Finally, from the single cIMS node availability in (4), we can derive the whole chain steady-state availability, viz.

$$A_{cIMS} = \prod_{j=1}^4 A_j. \quad (5)$$

The product in (5) takes into account the RBD scheme of Fig. 2, where, the series connection among nodes suggests that each single node must be working for the whole chain availability.

V. PERFORMABILITY MANAGEMENT

In this section we describe our performability management system which, for the SRN evaluation, takes advantage of TimeNET [23] tool. Denoting by setting S a generic cIMS deployment composed of a specific number of CNRs, our purpose is to solve an optimization problem, namely to find the optimal settings satisfying the high availability requirement at minimal cost. In the following numerical analysis, for each setting, we assume that a node can host a maximum of 4 CNRs, whereas a CNR can host a maximum of 6 containers. Given E_j the cost (expenditure) of node j (composed of h CNRs), the global cost of a cIMS setting amounts to

$$E(S) = \sum_{j \in \{P, S, I, H\}} E_j. \quad (6)$$

In practice, we look for S^* minimizing the overall cost while guaranteeing a given steady-state availability requirement A_0 . Denoting by $\mathcal{R} = \{S : A_{cIMS}(S) \geq A_0\}$ the space of possible settings which satisfy the steady-state availability constraint, the formal solution of the problem is:

$$S^* = \arg \min_{S \in \mathcal{R}} E(S). \quad (7)$$

To easily manage the optimization problem, we devise an algorithm (referred to as *OptChains*) that: *i*) provides automatic construction and evaluation of SRN models for different schemes and redundancy levels; *ii*) allows to extract (based on an exhaustive search with pruning logic) the interesting redundant configurations satisfying specific criteria such as “five nines” requirement and cost reduction. Before detailing such an algorithm, two assumptions have to be clarified. The first one concerns the diversity we impose between HSS and remaining containers (P, S, I), due to a more complex design that involves HSS hosting a database infrastructure. This criticality is taken into account by considering an improved robustness of HSS calling for an *extra* replica to be added for such node, thus, $\gamma_{HSS} = \gamma_{CSCF} + 1$ (to avoid hard notation, we will refer to γ in place of γ_{CSCF}). The second assumption pertains to the assignment cost criterion. In line with some top players (e.g. Microsoft Azure, Amazon AWS) which apply a policy pricing “per-layer” (e.g. hardware, OS, hypervisor, licenses, etc.), we consider three dimensionless cost contributions (each one normalized to 1 for simplicity) per CNR: *i*) a cost per container where software logic and licenses costs are included; *ii*) a cost per docker/VM which embodies OS costs; *iii*) a cost per hardware/hypervisor which includes infrastructure costs.

Algorithm 1: OptChains

```

1 Initialize the vector R containing all possible settings;
2 for settings  $\in$  R do
3   if  $r_i == 1$  given  $\gamma$  then
4     Ext. Call TimeNET (R)
5     if  $Av \geq A_0$  then
6       R  $\leftarrow$  R ( $A_0$ )
7     end
8   end
9 end
10 Input: R ( $A_0$ ),  $\gamma$ , max_cont,  $g_1$ ,  $g_2$ ,  $g_3$ ,  $G$ , minCost=inf;
11 for  $p \in R_{pcscf}(A_0)$  do
12   calculate  $E_p$ 
13   if  $E_p > g_1 \cdot \text{minCost}$  then
14     continue;
15   end
16   for  $s \in R_{scscf}(A_0)$  do
17     calculate  $E_s$ 
18     if  $E_p + E_s > g_2 \cdot \text{minCost}$  then
19       continue;
20     end
21     for  $i \in R_{iscsf}(A_0)$  do
22       calculate  $E_i$ 
23       if  $\sum_{k=p,s,i} E_k > g_3 \cdot \text{minCost}$  then
24         continue;
25       end
26       for  $h \in R_{hss}(A_0)$  do
27         calculate  $E_h$ 
28         if  $\sum_{k=p,s,i,h} E_k > G \cdot \text{minCost}$  then
29           continue;
30         end
31         save [Setting  $S$ ,  $Av(S)$ ,  $E(S)$ ]
32       end
33     end
34   end
35 end

```

A. The OptChains Algorithm

The proposed algorithm (*OptChains*) can be split in two parts (see pseudo-code). The first one (lines 1-9) embeds a procedure that allows to: *i*) automatically build and evaluate SRNs for different settings and with different redundancy through a call to TimeNET; *ii*) keep settings with reward rate equal to 1 (given γ) guaranteeing “five nines” requirements.

In the second part (lines 10-35) the algorithm accepts as input the obtained reduced number of settings, and embodies a procedure based on a pruned exhaustive search. Actually, input includes γ value, the maximum number of containers (*max_cont*) permitted per CNR, and 4 weight factors (g_1 , g_2 , g_3 , and G) useful to tune the pruning process. Initially, the exhaustive search allows to evaluate costs for all admissible settings per node, by starting from the P-CSCF (line 11) through a cycle on the sub-vector $\mathbf{R}_{pcscf}(A_0)$. Settings exhibiting a cost greater than g_1 times the cost of entire cIMS series are *pruned*, thus, the algorithm passes to the S-CSCF exploration (line 16). Similarly, settings with a cost greater than g_2 times the cost of entire cIMS series are pruned. The sub-procedure is then repeated also for I-CSCF (line 21) and HSS (line 26). It is worth noting that, G factor at lines 28

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	$2000 \leq \text{IMS sessions} \leq 5000$
c	performance capacity	1000 IMS sessions
A_0	steady-state availability requirement	0.99999

means that $G\%$ more settings are kept for backup purposes. Finally, algorithm saves a 3-tuple vector (line 31) containing replica distributions for the 4 cIMS nodes, along the pertinent availability value and cost.

The choice of weight factors can follow any reasonable criterion. In our case, we assume that each of 4 cIMS nodes is worth 0.25 of the whole cIMS deployment. Now, at beginning of procedure, we start by pruning all settings whose P-CSCF cost is greater than its redoubled value. This conservative choice is equivalent to select $g_1 = 0.5$ (in place of 0.25). Similarly, when the procedure goes ahead, the choice of settings to prune is performed according the re-scaled factors $g_2 = 0.75$ and $g_3 = 1$. Finally, we assign $G = 1.15$ indicating that we keep an extra amount (15% more) of settings in case the designer would have more options than nominal ones.

With such choices, and, by adopting for instance $\gamma = 4$, *OptChains* algorithm allows to reduce the number of admissible cIMS settings from an order of magnitude of 10^{13} , derived by considering 7 values (0-6) varying across 4 nodes and then composing a chain of 4 elements $((7^4)^4)$, to a more manageable 10^5 order of magnitude. On a PC equipped with Intel Core CPU i5 – 3230@2.60GHz and with a RAM of 8 GB, the execution time of the algorithm amounts to about 330 s (excluding the external call to TimeNET tool).

B. Numerical Analysis

In the advanced numerical analysis, we consider a variable performance demand $W \in [2000, 5000]$ IMS sessions, and a performance capacity (again in terms of IMS sessions) $c=1000$, by assuming, without loss of generality, $c_j = c$. With these assumptions, a provider which has to guarantee up to 3000 concurrent IMS sessions, for instance, needs to use at least 3 containers on top of each deployed CNR, thus, $\gamma = 3$. Parameters values are reported in Table I, where, due to lack of measured data on field, most of MTTFs and MTTRs are derived by [14], which, in turn, relies on “cluster data trace” logging events released by Google.

Let us start to analyze some settings obtained as output of the automated procedure. For the sake of simplicity, we report in Table II a selection of 4 (among 1000) settings $(S_1, \dots, S_4)$. The first column indicates the value of γ that, for this analysis, is supposed to range between 2 and 5. Let us now clarify the notation adopted for each CNR (columns 3-6 of Table II). In case of $\gamma = 2$, setting S_1 , P-CSCF node, $([2 \ 2 \ 0 \ 0])$

TABLE II: A selection of 4 exemplary settings (S_1, S_2, S_3, S_4) with different distributions of CNRs grouped per certain γ .

γ	Setting	P-CSCF	S-CSCF	I-CSCF	HSS	# CNT
2	S1	[2 2 0 0]	[3 1 1 0]	[2 2 0 0]	[3 3 0 0]	19
	S2	[2 2 0 0]	[3 2 0 0]	[2 2 0 0]	[3 3 0 0]	19
	S3	[2 2 0 0]	[2 2 1 0]	[2 2 0 0]	[3 3 0 0]	19
	S4	[2 2 2 0]	[2 2 0 0]	[2 2 0 0]	[3 4 0 0]	21
3	S1	[3 3 0 0]	[3 3 0 0]	[3 3 0 0]	[6 3 1 0]	28
	S2	[3 3 0 0]	[4 3 1 0]	[3 3 0 0]	[4 4 0 0]	28
	S3	[3 3 1 0]	[3 5 0 0]	[3 3 0 0]	[4 4 0 0]	29
	S4	[3 2 2 1]	[3 3 0 0]	[3 3 0 0]	[4 4 0 0]	28
4	S1	[4 6 0 0]	[2 2 3 0]	[2 2 4 0]	[5 4 1 0]	35
	S2	[6 4 0 0]	[4 4 0 0]	[4 4 0 0]	[5 5 0 0]	36
	S3	[5 5 0 0]	[4 4 0 0]	[4 4 0 0]	[5 5 1 0]	37
	S4	[2 2 2 2]	[4 4 0 0]	[4 6 0 0]	[5 5 0 0]	36
5	S1	[5 5 0 0]	[5 5 0 0]	[5 5 0 0]	[5 5 1 0]	41
	S2	[5 5 0 0]	[3 3 2 2]	[6 6 0 0]	[6 6 0 0]	44
	S3	[5 5 0 0]	[5 5 0 0]	[5 4 1 0]	[4 3 3 3]	43
	S4	[3 3 3 5]	[5 5 0 0]	[5 5 0 0]	[6 6 0 0]	46

indicates that 2 out of 4 CNRs (of P-CSCF type) are used, and remaining twos are set to 0; furthermore 2 containers per CNR are used. Similarly, for the S-CSCF $([3 \ 1 \ 1 \ 0])$, 3 out of 4 CNRs (of S-CSCF type) are exploited, and remaining one is set to 0; moreover, first CNR hosts 3 containers and the remaining twos host 1 container, respectively. Finally, the last column of Table II reports the total number of containers deployed for each setting.

The adopted notation turns out to be useful to easily assign the cost E per setting. For example, the cost of setting S_1 for $\gamma = 2$ can be determined as follows: E_P (cost for P-CSCF) amounts to $1 \cdot 4$ (CNT) $+ 1 \cdot 2$ (DCK+VM) $+ 1 \cdot 2$ (HPV+HW); E_S (cost for S-CSCF) amounts to $1 \cdot 5$ (CNT) $+ 1 \cdot 3$ (DCK+VM) $+ 1 \cdot 3$ (HPV+HW); E_I (cost for I-CSCF) amounts to $1 \cdot 4$ (CNT) $+ 1 \cdot 2$ (DCK+VM) $+ 1 \cdot 2$ (HPV+HW); E_H (cost for HSS) amounts to $1 \cdot 6$ (CNT) $+ 1 \cdot 2$ (DCK+VM) $+ 1 \cdot 2$ (HPV+HW). Accordingly, the total cost of S_1 for $\gamma = 2$ amounts to $E = E_P + E_S + E_I + E_H = 37$. It is now interesting to analyze the relation between cost and availability constraints for some desired performance levels γ . At this aim, in Fig. 4 we show steady-state unavailability $1 - A_{cIMS}$ for the considered settings and values of γ . Each bar denotes the unavailability value per setting, whereas, the number inside it indicates the pertinent cost. The horizontal dashed line in correspondence of 10^{-5} is the “five nines” value that, if crossed, denotes a violation of high availability requirement. Let us analyze, for instance, the values for the case $\gamma = 2$. Setting S_1 exhibits an availability value of 0,999987 (the pertinent bar overcomes the horizontal dashed line) and a total cost of 37. On the contrary, S_2 (barely) satisfies the “five nines” constraint (with a value amounting to 0,99999006), but, with a lower cost (amounting to 35). Such apparently counterintuitive result (higher availability typically corresponds to stronger redundancy associated with higher costs) can be explained by noticing that, S-CSCF in setting S_2 hosts the same number of containers than S_1 , but, distributed among 2 CNRs (in lieu of 3) - see Table II. This, in turn, implies that, for S_2 , the cost of a whole CNR can be saved.

Different distributions of containers across settings can

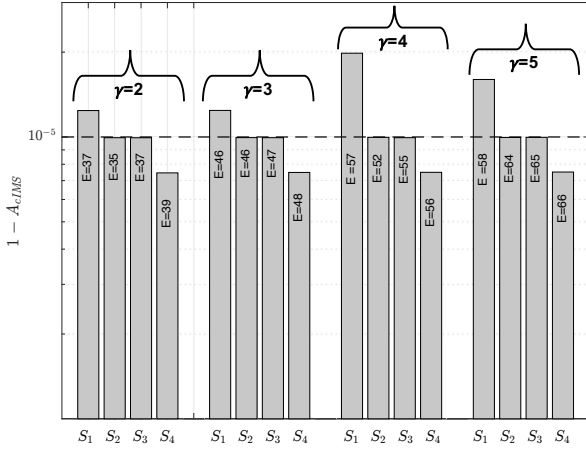


Fig. 4: Steady-state unavailability considering 4 exemplary settings (S_1, S_2, S_3, S_4) for: $\gamma = 2, 3, 4, 5$.

produce more pronounced uneven results. It is the case of $\gamma = 4$ case, where, setting S_1 exhibits an availability value of 0.99998017 with a cost amounting to 57, whereas, it is possible to satisfy high availability requirement by means of S_2 with a significantly lesser cost of 52. Moreover, a rather “safe” value of availability (0.99999251) is reached by S_4 with a slightly lesser cost amounting to 56. Similar considerations hold true for other settings and different values of γ , where, it is also possible to note that, as performance level grows, cost per each setting increases since the whole chain must afford a greater load in terms of cIMS concurrent sessions. In summary, such analysis reveals that, given a certain performance level to satisfy, a provider can decide to deploy a cIMS by considering multiple trade-offs involving: distribution of containers among deployed CNRs, costs, desired availability target.

VI. CONCLUSIONS

In this work we provide a performability assessment of a container-based IP Multimedia Subsystem (cIMS) infrastructure, considered a key architecture in the 5G ecosystem. First, we adopt a two-level hierarchy approach to model cIMS by involving two frameworks: Reliability Block Diagram (RBD) and Stochastic Reward Network (SRN). The former is useful to capture high level interconnections among cIMS nodes, whereas, the latter is amenable to characterize the probabilistic behavior of each node in terms of failure and repair events. Then, once defined the performance metric γ as a measure of cIMS to support a given load (in terms of number of sessions), we propose a performability assessment aimed at finding out the optimal set of configurations guaranteeing a desired performance level. Accordingly, we devise an algorithm (referred to as *OptChains*) that: *i*) provides automatic construction and evaluation of SRN models for different schemes and redundancy levels; *ii*) allows to extract (based on an exhaustive search with pruning logic) the interesting redundant configurations satisfying specific criteria such as “five nines” requirement and cost reduction.

REFERENCES

- [1] M. He, A. M. Alba, A. Basta, A. Blenk and W. Kellerer, “Flexibility in Softwarized Networks: Classifications and Research Challenges,” *IEEE Communications Surveys & Tutorials*, doi: 10.1109/COMST.2019.2892806, 2019.
- [2] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, M. F. Zhani, “Data Center Network Virtualization: A Survey,” *IEEE Communications Surveys Tutorials*, vol. 15, no. 2, pp. 909–928, 2013.
- [3] S. R. Chowdhury, M. A. Salahuddin, N. Limam and R. Boutaba, “Re-Architecting NFV Ecosystem with Microservices: State of the Art and Research Challenges,” *IEEE Network*, vol. 33, no. 3, pp. 168–176, 2019.
- [4] B. Spinnewyn, P. H. Isolani, C. Donato, J. F. Botero and S. Latré, “Coordinated Service Composition and Embedding of 5G Location-Constrained Network Functions,” *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1488–1502, 2018.
- [5] H. Moens and F. De Turck, “Customizable Function Chains: Managing Service Chain Variability in Hybrid NFV Networks,” *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 711–724, 2016.
- [6] B.R. Haverkort, R. Marie, G. Rubino, and K.S. Trivedi, *Performability modelling techniques and tools*. Chichester(UK), John Wiley and Sons, Ltd., 2001.
- [7] Implement a high availability architecture. Available online: https://www.ibm.com/garage/method/practices/manage/practice_high_availability/.
- [8] E. Jamhour, and M.C. Penna, “A state-based availability model to shared mesh protection in MPLS-TP networks with preemption support,” in *IEEE NOMS 2012*, pp. 926–932, 2012.
- [9] G. Su, and A. Iyengar, “A highly available transaction processing system with non-disruptive failure handling,” in *IEEE NOMS 2012*, pp. 409–416, 2012.
- [10] A. F. de la Cruz, J. P. Muñoz-Gea, P. Manzaneres-Lopez and J. Malgosa-Sanahuja, “Network failures support for traffic monitoring mechanisms in software-defined networks,” in *IEEE NOMS 2016*, pp. 691–694, 2016.
- [11] L. Zhang Y. Wang, X. Zhong, W. Li and S. Guo, “Resource-saving replication for controllers in multi controller SDN against network failures,” in *IEEE NOMS 2018*, pp. 1–7, 2018.
- [12] 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.
- [13] 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.
- [14] 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.
- [15] R. Matos, J. Dantas, J. Araujo, K.S. Trivedi, and P. Maciel, “Redundant Eucalyptus private clouds: Availability modeling and sensitivity analysis,” *J. of Grid Computing*, vol. 15, no. 1, pp. 1–22, 2017.
- [16] 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.
- [17] 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.
- [18] M. Di Mauro, G. Galatro, M. Longo, F. Postiglione, and M. Tambasco, “IP Multimedia Subsystem in a containerized environment: availability and sensitivity evaluation”, in *IEEE PVE-SDN19/Netsoft*, (Paris, France, Jun. 2019).
- [19] 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.
- [20] 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.
- [21] Docker. Available online: <https://www.docker.com/>.
- [22] 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.
- [23] 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.